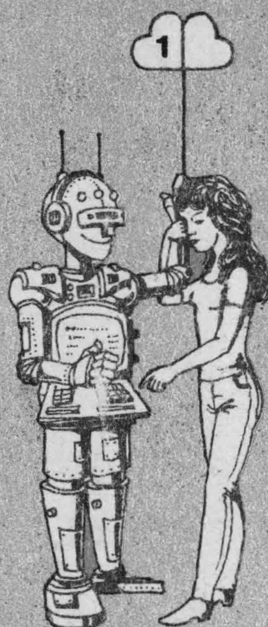
LIVIU DUMITRAȘCU

ÎNVĂȚĂM



MICROELECTRONICA INTERACTIVĂ

EDITURA TEHNICĂ

AUTOMATICĂ
INFORMATICĂ
ELECTRONICĂ
MANAGEMENT



BIBLIOTECA DE

Automatică–Informatică–Electronică–Management

SERIA INIȚIERE

E. VASILIU
INIȚIERE ÎN DISPOZITIVELE SEMICONDUCTOARE
D. STANOMIR
INIȚIERE ÎN ELECTROACUSTICĂ
W. TRUSZ
ABC-UL REPARĂRII RADIORECEPTOARELOR
Traducere din lb. polonă (Ciclul ABC-uri)
A. POPA
ABC DE PROTECȚIA MUNCII (ciclul ABC-uri)
MARGARETA DRĂGHICI
INIȚIERE ÎN COBOL
STELIAN NICULESCU
INIȚIERE ÎN FORTRAN
PAUL CONSTANTINESCU ȘI ZAHARIA NICOLAE
INIȚIERE ÎN ORGANIZAREA ȘI PROIECTAREA SISTEMELOR DE CONDUCERE
I. V. DUMITRESCU ș.a.
INIȚIERE ÎN TELEPRELUCRAREA DATELOR
I. CREȚU
INIȚIERE ÎN ESTETICA PRODUSELOR (Ciclul ABC-uri)
E. AISBERG
ABC DE RADIO ȘI TELEVIZIUNE
Traducere din limba franceză
J. D. WARNIER, B. MI FLANAGAN
INSTRUIRE ÎN PROGRAMARE
Traducere din limba franceză
I. H. BERNHARD, B. KNUPPERTZ
INIȚIERE ÎN TIRISTOARE
Traducere din limba germană
W. DEPPEERT, K. STOLL
INIȚIERE ÎN PNEUMOAUTOMATICĂ
Traducere din limba germană
E. VASILIU
INIȚIERE ÎN RADIOELECTRONICĂ CUANTICĂ
V. POPESCU
INSTRUIRE PROGRAMATĂ ÎN CALCULATOARE NUMERICE
ȘT. BÎRLEA
INIȚIERE ÎN CIBERNETICA SISTEMELOR INDUSTRIALE
I. PAPADACHE
AUTOMATIZĂRI INDUSTRIALE, INIȚIERE, APLICAȚII
ȘT. NICULESCU
FORTRAN, INIȚIERE ÎN PROGRAMARE STRUCTURATĂ
J. FORRESTER
PRINCIPIILE SISTEMELOR: TEORIE ȘI AUTOINSTRUIRE PROGRAMATĂ
Traducere din lb. engleză — S.U.A.
P. DRANSFIELD, D. F. HABER
INSTRUIRE PROGRAMATĂ ÎN METODA LOCULUI RĂDĂCINILOR
Traducere din lb. engleză — S.U.A.
R. BĂRSAN
DISPOZITIVE ȘI CIRCUITE INTEGRATE CU TRANSFER DE SARCINĂ
D. RODDY
INIȚIERE ÎN MICROELECTRONICĂ
Traducere din lb. engleză
NICULESCU Cl., IOSIF M.
INIȚIERE ÎN COMUNICAȚIILE PRIN FIBRE OPTICE
CSABAI DÁNIEL
TEHNICA SONORIZĂRII (traducere din lb. maghiară)
MITROFAN GH., PFLANZER G.
INIȚIERE ÎN TELEVIZIUNEA ÎN CULORI
RADU NEGOESCU
INIȚIERE ÎN ELECTRONICA BIOMEDICALĂ
RADU NEGOESCU
INSTRUMENTAȚIA ELECTRONICĂ BIOMEDICALĂ
A. PETRESCU ș.a.
TOTUL DESPRE... CALCULATORUL PERSONAL aMIC
L. DUMITRAȘCU ș.a.
ÎNVĂȚĂM FORTRAN... CONVERSÎND CU CALCULATORUL
L. DUMITRAȘCU
ÎNVĂȚAM COBOL... CONVERSÎND CU CALCULATORUL

Dr. ing. LIVIU DUMITRAȘCU

ÎNVĂȚĂM MICROELECTRONICA INTERACTIVĂ

TOTUL DESPRE... BASIC
ÎN 14 CONVERSAȚII ȘI 7 SINTEZE

1



EDITURA TEHNICĂ BUCUREȘTI, 1989

LIVIU DUMITRAŞCU a absolvit în 1970 Institutul Politehnic din Bucureşti. În anul 1986 a susţinut teza de doctorat în specialitatea automatizarea instalaţiilor petroliere. Este şef de lucrări la Institutul de Petrol şi Gaze Ploieşti. A activat timp de aproape zece ani la C.E.P.E.C.A. din cadrul Academiei „Ştefan Gheorghiu" unde a predat îndeosebi limbaje de programare în cadrul cursurilor pentru formarea specialiştilor în informatică. A absolvit în Franţa cursuri de specializare în informatică şi tehnică de calcul. A participat la realizarea în colectiv a unor produse-program şi sisteme informatice. A publicat singur sau în colaborare la Editura Academiei R. S. România şi Editura Tehnică mai multe articole şi cărţi, o carte editându-se şi în limba engleză. Cărţile sale din acelaşi ciclu de la Editura Tehnică s-au bucurat de o mare popularitate. Domenii de interes: generarea automată a programelor prin tabele de decizie, calculatoare personale, microbaze de date, inteligenţă artificială.

Prefaţă: ec. NICOLAE BADEA DINCĂ
Recenzii: dr. ing. NICOLAE ŢĂPUŞ
prof. dr. ing. ADRIAN PETRESCU

Redactor: ing. PAUL ZAMFIRESCU

Ideea graficii de copertă şi vignete: ing. GHEORGHE LUCHIAN
Execuţia coperţii: RADU GHEORGHIAN, SIMONA DUMITRESCU, BOGDAN ILIE
Desene: arh. CONSTANTIN MIHĂESCU
Tehnoredactor şi machetare: MARIA TRĂSNEA

ISBN 973-31-0010-2
ISBN 973-31-0011-0

Coli de tipar: 39. Bun de tipar: 12. VI. 1989.
C.Z. 621.38

Tiparul executat sub com. 260/1988 la
Întreprinderea Poligrafică „Crişana"
Oradea, str. L. Sălăjan nr. 105
Republica Socialistă România

PREFAȚĂ

... Să facem totul pentru înarmarea clasei muncitoare, a intelectualității, a întregii națiuni, în primul rînd a tineretului, cu cele mai înalte cunoștințe din toate domeniile...

NICOLAE CEAUȘESCU

(din cuvîntarea la Plenara Consiliului Național al Oamenilor Muncii, 13 iulie 1989)

Am acceptat solicitarea Editurii Tehnice de a scrie cîteva cuvinte de bun sosit acestei lucrări, cel puțin din următoarele 4 motive: – sînt lucrător științific la Institutul de Tehnică de Calcul și Informatică (I.T.C.I.) care are numeroase și variate preocupări în domeniul larg pe care îl intersectează cartea; – sînt (co)autor al unor cărți de profil, apărute și în curs de apariție; – sînt consilier al redacției de specialitate și – nu sînt nici coleg, nici prieten al autorului, așa încît puteți crede cu seninătate în obiectivitatea mea de prim-cititor, care nicicum – vă asigur – nu ar fi fost perturbată.

Oricum aș privi, separat din fiecare din cele patru ipostaze sau integrat, răspunsul meu la această editare este favorabil. Să mă explic:

– I.T.C.I. are ca obiective principale cercetarea științifică și ingineria tehnologică în domeniu, elaborarea și instalarea unor sisteme informatice, organizarea, dezvoltarea și desfășurarea industriei naționale de programe ș.a; organizăm cursuri teoretice și practice de formare, perfecționare și specializare a personalului ce lucrează în informatică și a utilizatorilor de tehnică de calcul; cooperăm – și prin unități distribuite teritorial – cu producătorii și beneficiarii de tehnică de calcul, desfășurînd cercetare și proiectare pentru diferite echipamente, pentru sisteme de operare, limbaje, programarea aplicațiilor etc.; asigurăm întreținerea, dezvoltarea și funcționarea Bibliotecii Naționale de Programe; motive, credem, suficiente, de a saluta apariția a cît mai multor cărți bune în domeniu, fie că ele sînt realizate de specialiștii institutului, fie, ca în acest caz, în alte nuclee de învățămînt-cercetare-producție; mai remarcăm, în contextul prefeței, preocupările speciale ale I.T.C.I. în informatica educațională; studiile de impact al calculatoarelor în educație, elaborarea și reproducerea industrială a programelor informatice pentru diverse discipline, cercetări pentru instruirea asistată pe calculatoare personale în BASIC și LOGO, realizarea de laboratoare didactice-fixe și mobile pentru predarea informaticii la diferite nivele, organizarea olimpiadelor de profil ș.a.; laboratorul mobil de instruire – INFOBUS – este apreciat de multe instituții de învățămînt din țară și va putea fi promovat în cadrul programelor UNESCO;

— de curînd a apărut o lucrare de automatică şi informatică aplicată, la care am adus o oarecare contribuţie privind metodologia şi tehnicile de programare, iar la porţile cititorilor va bate curînd un „ABC de calcul electronic" realizat de un colectiv de la I.P.B., I.T.C.I., Fabrica de calculatoare, C.N.O.P. etc., adresat cu deosebire tineretului, care se completează de minune cu conţinutul prezentei cărţi de programare;

— noul ciclu generic "Învăţăm microelectronica interactivă" pe care Editura Tehnică îl iniţiază este într-adevăr mult mai generos decît pot cuprinde aceste prime volume, care nu se preocupă de o multitudine de aspecte constructive, funcţionale ş.a. ale echipamentelor electronice interactive şi nici de componente specializate ale acestora ci doar de comunicarea cu ele, prin intermediul unui limbaj consacrat conversaţiilor om—maşină; apreciind această adevărată „enciclopedie BASIC" am pus la dispoziţia redacţiei documentaţii de mare actualitate;

— cititorul avizat mai remarcă atît stilul original cît şi organizarea funcţională a cărţii, în două volume cu planuri juxtapuse, în care se intersectează 10 direcţii judicioase, ce asigură un ansamblu de manuale practice de instruire şi aplicare a programării în BASIC, cu codificări pe numeroase clase de echipamente, cu exemple selecţionate şi cu un bogat conţinut de îndreptar ce extinde limitele stricte ale limbajului (microprocesoare, sisteme de operare, aplicaţii tipice, alte limbaje de nivel înalt prezentate comparativ etc.).

În esenţă, felicitînd pe autor şi redacţia de informatică şi tehnică de calcul a Editurii Tehnice pentru această realizare, recomand unor cercuri largi de cititori, utilizatori şi creatori de tehnică de calcul din toate ramurile economiei noastre să studieze aceste preţioase volume.

ec. NICOLAE BADEA DINCĂ
Director adjunct ştiinţific I.T.C.I.

N.R. Culoarea în carte s-a dorit funcţională, nu doar decorativă: temele din conversaţii şi soluţiile lor, modulele tip de analiză şi proiectare structurată a programelor din exemple, trimiterile în vol. 1 la imaginile programelor din vol. 2, sinteza 20, comenzile şi instrucţiunile explicate din sinteze, chiar şi ... vignetele simbolice şi unele titluri evidenţiate stau mărturie.



Celor mai dragi tineri,
ai mei, Daniel şi Andrei

Paşi printre... conversaţii şi sinteze.
Un dialog autor–editor–cititor (I)

☐ **Mai întîi... cititorul, "le maître"**. Vă explic titlul cu pătrăţel, mărturisindu-vă – discret şi modest – că, pe cînd începeam să scriu cartea, am dat răspunsul corect la o întrebare eternă: "ce a fost mai întîi, oul sau găina?", e drept într-un **caz** al meu, particular, carte/cititor; mă veţi întreba, poate, "în ce fel, şi, chiar de nu sînteţi curioşi, vă voi spune, deşi dezvălui – să nu zic vorbă mare – un scump secret: în cazul dat, am procedat simplu (ca bonjour), **tăind nodul gordian** şi, ce să vezi, ... a **ţipat la tăiat**; imediat am dedus logic "e cititorul, nu cartea" (e drept, şi cartea poate ţipa, cînd e anapoda, dar... să lăsăm altora calea asta). Şi, ca să nu fie o şaradă, să vă descriu **cazul**. Discutam, animat, cu redactorul, planul cărţii şi nimerim, ca din întîmplare, într-o frumoasă librărie, unde – fatalitate?! – ce auzim: " – N-aveţi cumva cartea «Învăţăm BASIC... conversînd cu... ?»" "– Nu", vine răspunsul sec. "– De ce oare, că a fost promisă în altă carte a aceluiaşi autor, «Învăţăm COBOL... etc.?»". "– Nu ştim, mai încercaţi prin vară... peste trei ani. Cărţile informatice se scriu mai greu şi apar mai rar". Mă înroşesc în mine (pentru toţi) şi intru în... conversaţie, să văd dacă cititorul ştie ce caută şi cum ar vrea să fie, dac-ar fi. Am **tăiat**, astfel, nodul... de care vă vorbeam, iar cititorul **a ţipat** la mine (q.e.d. – sau, ceea ce era de demonstrat – cititor **da**, carte **ba**).

"– Cum să nu ştiu, dom'le, zice, dar nu cumva dumneavoastră sînteţi niscaiva autori sau editori?". "– Nu sînt (încă autor..." zic, în gîndul meu), dar dacă am fi toţi trei ce ai vrea să facem?". "– Păi, mai întîi o **copertă** frumoasă, cu un **titlu** atrăgător, vorba lui Vlahuţă: "La titlu să iei seama: sonor, bombastic, vag / Şi cu ceva macabru să ne loveşti din prag; / Că nu-i zor nicidecum / Ca titlul şi povestea să aibă acelaşi drum", apoi un **cuprins...** cuprinzător, pe înţelesul **tuturor** (tuturor e un fel de a spune că şi pentru mine – care sînt muncitor specialist într-o secţie în care s-au instalat terminale conversaţionale – şi pentru fiul meu, elev la informatică şi candidat la facultate, şi pentru profesorii lui... măcar). Ce mai, cînd om lua-o în mînă, **cartea să vorbească singură** (că autorii sînt la treburile lor); să plecăm, deci, de la simplu la complicat şi să nu batem cîmpii decît în locuri ştiute, pe care cine vrea să le poată sări; fără prea multă teorie, dar nici în doi peri, să fie la obiect, cu programe alese, să ne ducă de la particular la general şi odată ajunşi aici să ştim oricînd să facem cale-ntoarsă. Ca să nu ne plictisim, să aibă culoare, să fie veselă, incitantă şi, mai ales, **să ne pună la treabă...** materia cenuşie, cerîndu-ne mereu să dezlegăm (sau să legăm) cîte ceva. Fie vorba între noi, planul cărţilor anterioare din ciclu, cu conversaţii şi sinteze, n-a fost rău, l-am putea păstra, făcîndu-l mai consistent". Aşa zise cititorul şi se duse, aşteptînd încrezător rezultatul.

Îngîndurat ("...era oare omul meu?! Da!" mi-am zis), mi-am adus aminte că şi redactorul (autor al unui BASIC şi şcolit informatic prin Franţa,

ca şi subsemnatul) mi-a vorbit de un personaj — şef şi executant în noul oficiu de calcul al tipografiei orădene — pe care l-a întîlnit prima oară lucrînd de zor (după o uşă cu "strict oprit") la un terminal, cu "Învăţăm COBOL..." pe masă, şi care, întrebat pe neaşteptate, a răspuns: "... E o carte **practică, învăţ şi lucrez după ea,** dar (era şi un dar) ... ar fi fost bine să aibă şi ..., dar dumneavoastră cine sînteţi, ... nu ştiţi să citiţi ... etc." De aceşti stăpîni (şi prieteni) ai noştri — cititorii — căutăm să ascultăm şi acum!

Am trecut, deci, cu redactorul, la proiectarea cărţii, ghidîndu-ne nu numai după auzite ci şi după altele, cum ar fi (bine-ar fi!) planul, viu în suflet, al grădinilor Palatului din Versailles (ici o arteziană, colo o Diană, ici un potir, colo un satir, ici un copac, colo un lac ... totul, însă, riguros, cu perspectivă carteziană).

☐ **Apoi, multe manuscrise, iteraţii şi, în fine, cartea.** Pe un **plan unitar** al fiecăreia din cele 14 conversaţii (cu exemple, teme, soluţii), "distilat" în cursul editării, am conceput **exemple şi programe de instruire,** care s-au închegat rapid într-o **concepţie structurată** de analiză—proiectare—programare în BASIC, redată sistematic, ca un laitmotiv, prin **documentaţii tipice** (diagrame de structură, pseudocod, scheme logice). În paralel, programele conversaţiilor au fost **codificate în variantele de BASIC** pe calculatoare personale, personal-profesionale, minicalculatoare, calculatoare medii; **cititorul a învăţat comenzile şi instrucţiunile uzuale,** salvarea, depanarea şi documentarea programelor, structurile fundamentale de programare (secvenţa, selecţia, iteraţia), crearea, actualizarea, consultarea şi listarea fişierelor etc. S-au rezolvat **mai toate tipurile de probleme** adecvate limbajului: calcule numerice, alfanumerice, vectoriale, matriciale, lucrul cu fişiere secvenţiale, cu acces direct, cu fişiere masive virtuale, grafică bi- şi tridimensională, lucrul cu meniuri, ferestre, culori, muzică, simularea mişcărilor, animaţie, conversaţiile încheindu-se cu un studiu de caz complex. Variante ale manuscriselor, programelor şi ale volumului 1 au fost supuse, înainte şi în timpul editării, unor **testări** îndelungi ... cu public, mai ales la I.P.G. Ploieşti, I.R.E. Ploieşti, I.P.R.S. Băneasa.

Sintezele, ce ocupă volumul 2, au fost elaborate ulterior (cu excepţia sintezei 20, cu programele exemplelor din conversaţii) ţinînd seama de observaţiile din testări/recenzări şi după consultarea unei bogate bibliografii, în bună parte recomandată de redacţie. **Ele sînt construite în alt mod decît conversaţiile,** constituind un mediu compact de consolidare, completare şi regăsire a cunoştinţelor, cît şi de extindere a lor asupra componentelor, echipamentelor, software-ului de sistem, aplicaţiilor, programării, în general, şi programării în BASIC, în special. **Paşii** (II şi III) **printre conversaţii şi sinteze** (pag. 606, v.1 şi pag. 8, v.2) lămuresc, pe de o parte, cele zece direcţii ce se intersectează în carte, iar, pe de altă parte, tratează noutăţi ale domeniului, cunoscute în ultimele faze editoriale.

☐ **Mulţumiri (măcar ale mele) transmit tuturor.** Dar, mai ales, următorilor: prof. dr. ing. Ioan Dumitrescu, prof. dr. ing. Stelian Dumitrescu, conf. dr. Miron Oprea, prof. dr. Eugeniu Niculescu-Mizil, dr. ing. Adrian Davidoviciu, dr. ing. Valerie Marinescu, ing. Ion Trandafir, Constanţa Figaro, Florenţa Costea, dr. ing. Alexandru Ioachim, mat. Cristian Marinoiu, ing. Victor Stoicea, ing. Marius Manea, ing. Dan Drăgan, prog. Geo Voican, soţiei mele Livia Dumitraşcu, ec. Claudia Dulgheru, Leonida Băietu, conducerii Editurii Tehnice şi, de fapt, întregului său colectiv, cum şi Poligrafiei orădene „Crişana" (ing. şef Vladimir Kagan, şef producţie Rozalia Farkaş şi colectivul respectiv de maiştri şi muncitori tipografi).

Aştept verdictul cititorilor!

SUMAR
pentru volumele 1 și 2

VOLUMUL 1

Totul despre BASIC în 14 conversații și . . .

VOLUMUL 2

. . . 7 sinteze

CUPRINS DETALIAT

VOLUMUL 1

Totul despre BASIC în 14 conversaţii şi . . .

*** N.A. Pașii I, II, III provin (ca și . . . altele) și din inspirația și activitatea prietenului meu, ing. Paul Zamfirescu.**

TOTUL DESPRE BASIC ÎN 14 CONVERSAŢII ŞI...

CONVERSAȚIA 1

Editarea unui text (titlul cărții) cu calculatorul. Cel mai simplu program în BASIC. Metode pentru introducerea, adăugarea și corectarea interactivă a unei linii program. Comenzi BASIC: RUN și LIST



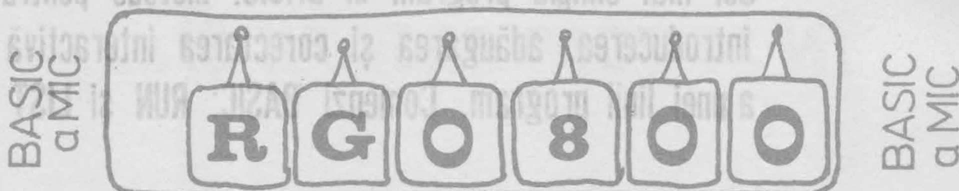
□ Imaginea programului BASIC-aMIC

vol. 2, pag. 198

Aprobare pentru un program-READY

Iată cît de simplu se pot întîlni două bucurii. Una a dumneavoastră care doriţi să învăţaţi BASIC... conversînd cu calculatorul şi una a noastră, care ne-am angajat să vă ajutăm, scriind această carte accesibilă tuturor vîrstelor. Cît de prietenoase sînt calculatoarele vă veţi convinge chiar din prima conversaţie, pe care vă invităm să o abordaţi urmînd cu atenţie şi întocmai textul. Apoi veţi vedea şi singuri ce va urma.

Treziţi, vă rugăm, geniul electronic adormit! Introduceţi de la claviatura aMIC-ului dvs. următoarea succesiune de tastări: **RGØ8ØØ***. Acţio-



naţi apoi tasta **[RETURN]** şi urmăriţi ce se afişează pe ecranul televizorului:

```
READY
■
```



Mesajul **READY** indică faptul că sistemul aşteaptă să interveniţi. El spune de fapt "sînt gata, este rîndul dvs. să faceţi ceva" sau, altfel zis: "am înregistrat comanda şi sînt gata să primesc informaţiile de la tastatură".

Pătrăţelul de sub **READY** care se deplasează pe ecran în timp ce dvs. tastaţi litere/cifre/caractere speciale (şir de caractere) este cursorul. El vă permite să ştiţi exact unde va fi tipărit următorul caracter. El spune de fapt "aici este locul unde va apare pe ecran următorul caracter". Dacă apăsaţi pe bara de jos cursorul se va deplasa fără să tipărească nimic.

Introducerea unei linii program

Modul de lucru cu calculatorul personal aMIC (pentru a-l determina să facă ceva pe cont propriu) este acela de a scrie pentru el un program, de a introduce acest program în calculator şi de a determina apoi calculatorul să execute programul introdus. Un program este o listă de instrucţiuni pe care calculatorul le înţelege şi le execută singur.

---

* Caracterul "Ø" reprezintă cifra zero.

Cu aMIC-ul puteţi conversa în BASIC, recunoscut pentru uşurinţa cu care poate fi înţeles, învăţat şi folosit.

Înainte ca orice program să poată fi parcurs sau executat, el trebuie introdus în memoria calculatorului, fie direct de la claviatură, fie cu ajutorul unei casete sau al unei dischete.

În această conversaţie vom considera că programele sînt introduse direct de la claviatură. Programele pot fi introduse în calculator numai atunci cînd acesta se află în stare aptă de operare. Aceasta înseamnă că introducerea unui program poate începe numai după apariţia pe ecran a mesajului **READY** urmat de simbolul cursor.

Pentru dvs., "experţii", care deja aţi parcurs prima conversaţie, aceste lucruri vă sînt familiare. Pentru aceia dintre dvs. care acum aţi început să conversaţi, să încercăm împreună introducerea şi execuţia unui program BASIC care să afişeze titlul cărţii pe care o lecturaţi:

"INVATAM BASIC . . .

. . . CONVERSIND CU CALCULATORUL"

Să introducem, pentru început, prima linie a programului care să editeze doar "INVATAM BASIC . . .". Tastaţi următoarea linie:

10 **PRINT** "INVATAM BASIC . . ."

Nu acţionaţi încă tasta **[RETURN]**!

Ştergerea caracterelor unei linii program

Dacă faceţi o greşeală este uşor să o corectaţi, mult mai uşor decît la o maşină de scris. Folosiţi tasta **[DEL]**. Atunci cînd aţi greşit, ştergeţi cu **[DEL]** totul înapoi pînă la caracterul greşit (inclusiv), după care îl retastaţi. Acum, examinaţi cu atenţie ceea ce aţi tipărit:

a) după cuvîntul **PRINT,** aţi inclus totul între ghilimele?

b) există cumva ghilimele suplimentare?

Dacă totul este în ordine, apăsaţi pe **[RETURN].**

Număr de linie

Priviţi, vă rugăm, acest program BASIC. Să ne obişnuim cu „imaginea" lui. El este alcătuit dintr-o singură linie, numerotată cu 10. Un program BASIC este compus dintr-o serie de instrucţiuni ce se execută într-o anumită secvenţă (ordine). Din acest motiv, liniile programului se vor numerota. Deoarece în BASIC-aMIC noi nu putem scrie decît o instrucţiune pe fiecare linie, putem enunţa prima regulă:

Regula 1. **Într-un program BASIC orice linie trebuie să înceapă cu un număr de linie.**

Numerele de linie sînt numere întregi pozitive, cuprinse între zero şi un număr destul de mare. Cel mai mare număr de linie posibil pentru calculatorul aMIC este 32767.

Editare şir de caractere: PRINT

Instrucţiunea **PRINT,** în engleză "tipăreşte", ce urmează după numărul de linie (10) realizează editarea şirului de caractere: "INVATAM

BASIC ..." plasat între ghilimele. Un şir de caractere poate să includă cifre (0; 1; .... 9), litere (A; B; ...) sau caractere speciale (+; –; *; ↑; ?; ...). Cînd ghilimelele definesc începutul şi sfîrşitul unui şir de caractere, ele nu pot fi folosite drept caractere în cadrul şirului.

Din acest exemplu reţineţi o nouă regulă:

Regula 2. **Fiecare număr de linie trebuie să fie urmat de o instrucţiune.**

Remarci

- La calculatorul personal aMIC nu aveţi nevoie de tasta **[SHIFT]** pentru obţinerea caracterelor majuscule, deoarece scrierea se face numai cu litere mari. Această tastă este necesară pentru tipărirea semnelor desenate deasupra tastelor (prin apăsarea simultană a tastei **[SHIFT]** şi a tastei cu semnul dorit).
- Cînd tipăriţi caractere sau spaţii, cursorul se mişcă spre dreapta.
- Calculatorul, singur, nu are de unde să ştie cînd aţi terminat o linie program; trebuie ca dvs să-i spuneţi acest lucru. La aMIC semnalul de terminare constă în apăsarea tastei **[RETURN]**.
- Dacă, totuşi, găsiţi o greşeală după ce aţi tastat o linie de program şi aţi apăsat **[RETURN]** nu puteţi folosi pentru corecţie tasta **[DEL]**. Deocamdată, vă recomandăm retastarea întregii linii peste ea însăşi.
- În BASIC este posibilă "scrierea peste" orice linie a programului prin simpla tipărire a numărului de linie şi introducerea versiunii revizuite a instrucţiunii.

Comanda de execuţie RUN

Să nu uităm de simbolul cursor. Pare a vă întreba: "Bine, ce urmează?".

Un program – chiar şi unul construit dintr-o singură linie – nu este de prea mare folos înainte de a fi executat. În BASIC, un program este executat imediat după ce se tipăreşte comanda **RUN** şi se apasă pe tasta **[RETURN]**.



Deci, tastaţi **RUN** şi apăsaţi pe **[RETURN]**. Dacă nu aţi făcut nici o greşeală, pe ecranul televizorului veţi putea citi:

```
RUN
INVATAM BASIC...
ERROR 01 IN LINE 10
READY
■
```

Felicitări! Primul dvs. program în BASIC este acum executat, dar ...

Un mesaj de eroare

Nu vă alarmaţi! Pentru a înţelege ce s-a întîmplat vă propunem să analizăm împreună rezultatul execuţiei programului introdus. Cînd ne-am adresat calculatorului cu instrucţiunea **PRINT** urmată de "ÎNVĂŢĂM BASIC..." i-am cerut să reproducă şirul de caractere plasat între ghilimele. După cum constataţi, pe ecranul televizorului s-a tipărit: ÎNVĂŢĂM BASIC dar ghilimelele nu au fost afişate! Pe linia imediat următoare a apărut însă ceva ciudat, un mesaj de eroare spunem noi:

ERROR 01 IN LINE 10

Acest mesaj este rezultatul unei erori interne săvîrşite de noi. Ne ajută să aflăm cînd am realizat un lucru greşit, eronat sau un lucru bun dar la timpul nepotrivit. Codul 01 din linia 10 (de altfel, singura linie din program) ascunde ceva! Ce anume? Consultînd lista cu mesaje de erori (vol. 2, pag. 2) a aMIC-ului, în dreptul codului 01 găsim: „programul nu se termină cu instrucţiunea **END** sau **STOP**".

Aici este momentul oportun pentru introducerea unei instrucţiuni BASIC speciale, instrucţiunea **END.**

Unde se sfîrşeşte programul? Instrucţiunea END

Sfîrşitul programului este ultima instrucţiune pe care dorim s-o execute calculatorul.

Este probabil evident că instrucţiunea **END** spune calculatorului că s-a atins punctul final al programului respectiv. În mod normal, executarea unui program ce nu conţine instrucţiunea **END** produce acelaşi rezultat ca şi în cazul cînd programul conţine instrucţiunea **END.**

Foarte mulţi îşi fac o obişnuinţă din a încheia fiecare program cu instrucţiunea **END,** deşi nu este întotdeauna necesar să se procedeze astfel. Regulile ce dictează utilizarea lui **END** în majoritatea dialectelor BASIC impun ca **END** să fie ultima instrucţiune din program. De obicei, lui **END** i se dă numărul 99 sau 999 sau 9999, în funcţie de cel mai mare număr de linie acceptat de calculator.

Interpretorul BASIC-aMIC cere ca programul să se termine obligatoriu cu una din instrucţiunile **END** sau **STOP.** În caz contrar, calculatorul va afişa mesajul de eroare cu codul 01, ceea ce s-a şi întîmplat, de altfel, în cazul programului nostru.

Cum să extindem un program?

Din cauza apariţiei mesajului de eroare cu codul 01, sîntem nevoiţi să lărgim vechiul program (existent în memorie) cu o nouă linie, cea a instrucţiunii **END.** În BASIC fiecare linie a programului trebuie să aibă un număr, calculatorul executînd programul în deplină concordanţă cu ordinea numerelor liniilor, începînd cu cel mai mic număr de linie şi terminînd cu cel mai mare număr de linie. Cum **END** trebuie să fie ultima instrucţiune din program, vă propunem să tastaţi:

99 END

Verificaţi cu atenţie dacă aţi introdus corect noua linie, după care tastaţi [RETURN]. Dacă totul este bine, tastaţi **RUN** şi apăsaţi pe **[RETURN]**. Sperăm că, de această dată, aţi obţinut un rezultat fără erori.

```
RUN
INVATAM BASIC...
READY
■
```

Felicitări! Primul dvs. program v-a reuşit! După această experienţă este bine să reţineţi următoarele două reguli importante.

Regula 3. **Calculatorul execută un program în deplină concordanţă cu ordinea numerelor liniilor, începînd cu cel mai mic număr şi terminînd cu cel mai mare număr de linie.**

Regula 4. **Liniile programului nu trebuie introduse neapărat în ordine crescătoare. Calculatorul le va executa însă în ordine crescătoare.**

Mai multe despre comanda de execuţia RUN

Acest program simplu, cu două linii, poate fi executat ori de cîte ori dorim, tipărind comanda **RUN** şi apăsînd pe tasta **[RETURN]**. Încercaţi să executaţi programul de trei ori sau de cîte ori doriţi. Veţi obţine, evident, acelaşi rezultat: INVATAM BASIC . . . , o singură dată.

După studierea acestei mici demonstraţii, este bine să remarcaţi că atîta timp cît programul rămîne intact în memoria calculatorului, el poate fi executat ori de cîte ori dorim.

După cum aţi putut constata, spre deosebire de liniile program, comanda **RUN** se introduce fără număr de linie, iar după apăsarea tastei **[RETURN]** calculatorul execută imediat această comandă. Liniile de program sînt precedate de un număr de linie, iar după acţionarea tastei **[RETURN]** calculatorul reţine doar numărul de linie şi linia respectivă, fără ca instrucţiunea (entitatea) din linia respectivă să fie executată.

Revenind la instrucţiunea PRINT

Cînd ne-am adresat calculatorului cu instrucţiunea **PRINT** i-am impus reproducerea şirului de caractere inclus între ghilimele. Vă puteţi întreba, pe bună dreptate, dacă expresia plasată între ghilimele nu trebuie să aibă neapărat un sens! Răspunsul este nu! Ea poate fi orice combinaţie de caractere, inclusiv spaţii, cu o singură excepţie – ghilimelele.

Instrucţiunea

10 **PRINT** "1+2=4"

"merge" la fel de bine ca şi instrucţiunea

10 **PRINT** "INVATAM BASIC . . ."

Deci, chiar dacă expresiile cuprinse între ghilimele nu au nici un sens, calculatorul va tipări exact ceea ce i se indică!

Primul test

Să părăsim pentru un moment programul BASIC şi să ne concentrăm atenţia asupra unui mic test. Verificaţi dacă puteţi răspunde la întrebările puse, înainte de a citi explicaţiile noastre.

1. Se consideră următorul program BASIC (pe care vă rugăm să nu-l introduceţi în calculator).

```
10 PRINT "LA MULTI ANI FRUMOSI!"
20 END
```

a) Precizaţi numerele de linie ale programului.

. . . . .

10
20

b) Precizaţi instrucţiunea care indică sfîrşitul programului.

. . . . .

END

c) Care este rolul ghilimelelor din cadrul instrucţiunii **PRINT** (linia 10).

. . . . .

Ghilimelele spun calculatorului că şirul de caractere începe cu litera L şi se termină cu caracterul special "!"

d) Ce se va afişa pe ecranul televizorului în momentul execuţiei programului?

. . . . .

```
LA MULTI ANI FRUMOSI!
READY
■
```

e) Dacă aţi executa programul de o mie de ori, ce aţi constata?

. . . . .

Se obţine acelaşi rezultat.

f) Precizaţi rezultatul execuţiei programului anterior în cazul cînd aţi introduce numai linia 10.

. . . . .

```
LA MULTI ANI FRUMOSI!
ERROR 01 IN LINE 10
READY
■
```

2. Cum va răspunde calculatorul aMIC în urma execuţiei următoarelor programe?

a)
```
10 PRINT " "
30 END
```

. . . . .

Un spaţiu, urmat de mesajul **READY**.

b)
```
10 PRINT "5–3"
20 END
```

. . . . .

5–3, urmat de mesajul **READY**.

c)
```
10 PRINT "AB# "
20 END
```

. . . . .

AB#

d)
```
10 PRINT "2>3"
20 END
```

. . . . .

2>3 (!)

INIT şterge ecranul

O cantitate excesivă de informaţii tipărite la televizor vă poate deruta la un moment dat. În consecinţă, se impune o curăţire a ecranului, fără

a afecta însă programul din memoria calculatorului. Acest lucru se poate realiza, în BASIC-aMIC,.cu instrucţiunea **INIT**.

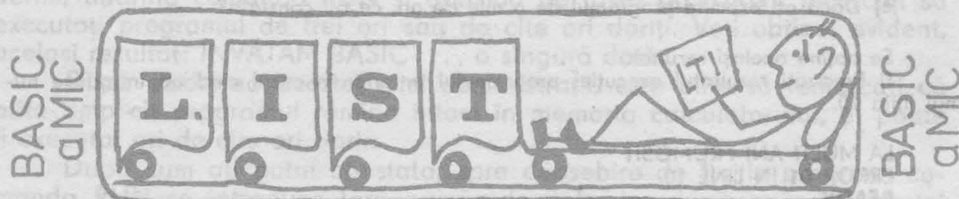
Este oare prea tîrziu pentru a adăuga această instrucţiune? Nu, bineînţeles că nu. Nu este niciodată prea tîrziu! Instrucţiunea **INIT** poate fi inserată în programul nostru înaintea liniei 10 prin tipărirea liniei

5 **INIT [RETURN]**

O nouă linie a programului poate fi inserată în orice loc în cadrul programului şi în orice moment, atîta timp cît sistemul este **READY**. Dvs. trebuie să alegeţi pur şi simplu un număr de linie, astfel încît instrucţiunea introdusă să "cadă" bine, adică acolo unde trebuie. În cazul nostru, ideea a fost de a introduce operaţia de ştergere a ecranului la începutul programului, aşa că numărul de linie poate fi orice cifră între 0 şi 9 inclusiv (noi am ales 5). Faptul că numărul de linie ce precede instrucţiunea **INIT** este mai mic decît numărul de linie ce precede pe **PRINT** ne dă certitudinea că INVATAM BASIC.... va fi întotdeauna tipărit după curăţirea ecranului şi înainte de a se fi executat instrucţiunea **END**.

Cînd scrieţi un program BASIC nu este nevoie să introduceţi liniile de program în secvenţă, adică în ordine crescătoare. Calculatorului nu îi pasă dacă instrucţiunile programului i se transmit într-o "ordine" aiurea. El urmăreşte numărul liniei şi nu ordinea în care am introdus noi instrucţiunile.

Pentru a vă lămuri singuri, cereţi calculatorului să afişeze întreg programul rezident în memorie.



Cu această ocazie, faceţi cunoştinţă cu o nouă comandă BASIC – **LIST**.

Revederea liniilor din program – comanda LIST

Comanda **LIST** permite afişarea programului introdus de către dvs. (tot programul sau numai anumite secvenţe) în memoria calculatorului.

Tastaţi:

LIST [RETURN]

Ce constataţi?

```
LIST
5 INIT
10 PRINT "INVATAM BASIC..."
99 END
```

Comanda **LIST** a confirmat faptul că sistemul a acceptat inserarea liniei 5 (şi desigur 99). În plus, remarcaţi modul în care au fost afişate liniile programului – în ordinea crescătoare a numerelor de linie.

Transmiteţi calculatorului comanda **RUN**, tastaţi **[RETURN]** şi analizaţi noul rezultat obţinut.

Este altceva, nu-i aşa? Într-adevăr, instrucţiunea **INIT** a şters cu "buretele" tot ecranul dar nu şi programul dvs. Dacă nu sînteţi convinşi, daţi din nou comanda **LIST**.

O sugestie

Faptul că într-un program pot fi scrise noi linii explică de ce am ales pentru instrucţiunile **PRINT** şi **END** numerele de linie 10 şi 99. Este întrutotul posibilă scrierea programelor cu numere de linie 1, 2, 3 şi aşa mai departe dar, procedînd astfel, nu am mai fi putut lăsa loc liber pentru introducerea de noi linii; şi veţi rămîne surprinşi cînd veţi vedea cît de des veţi dori să introduceţi într-un program noi linii. De aceea, vă recomandăm să lăsaţi spaţiu între fiecare număr de linie. Numerotarea liniilor cu multipli de 10 este probabil cea mai obişnuită practică, dar şi intervalele de 5 sau 20 sînt la fel de bune.

Cînd linia program depăşeşte 30 de caractere

Pentru aceia dintre dvs. care v-aţi amintit că aţi lăsat robinetul deschis acasă, ne putem opri aici (înainte de a pleca vă rugăm, totuşi, să "opriţi" calculatorul şi televizorul pentru a nu consuma energie electrică!). Alţii dintre dvs. au devenit deja pasionaţi de programare şi vor dori să-şi continue instruirea cu programarea afişării următorului rînd al titlului cărţii "... CONVERSÎND CU CALCULATORUL". (Noi nu ne asumăm responsabilitatea dacă întîrziaţi la masa de seară!).

Pentru a tipări cel de-al doilea şir de caractere ("... CONVERSÎND CU CALCULATORUL") va trebui să introducem o nouă linie în programul rezident în memoria calculatorului:

20 PRINT "... CONVERSIND CU CALCULATORUL"

Începeţi, vă rugăm, tastarea instrucţiunii. Aţi ajuns la situaţia din fig. de mai jos?

```
20 PRINT "... CONVERSIND CU CAL
CULATORUL"
```

Nu vă alarmaţi! Continuaţi să introduceţi şi restul caracterelor, după care nu uitaţi să apăsaţi pe **[RETURN]**.

Ce s-a întîmplat? După cum aţi putut constata linia program nu coincide cu linia de pe ecranul televizorului.

Lungimea liniei de pe ecranul televizorului este de 30 de caractere, în cazul nostru pînă la litera L inclusiv. Restul caracterelor (atenţie la noua poziţie a cursorului) se introduc automat pe linia următoare, fără a fi nevoiţi să mai tastăm **[RETURN]**.

Fizic, linia program reprezintă totalitatea caracterelor cuprinse între simbolul emis de calculator (cursor) şi terminatorul unei linii **[RETURN]** şi nu coincide neapărat cu o linie de pe ecranul televizorului. O linie program poate avea maximum 252 de caractere.

Nu sînteţi curioşi să vedeţi ce aţi realizat? Dacă da, tastaţi **RUN** şi **[RETURN]**. Trebuie să obţineţi următoarea imagine:

```
INVATAM BASIC...
...CONVERSIND CU CALCULATORUL
READY
■
```

Cum vi se par cele două şiruri de caractere pe care le-a afişat calculatorul dvs.? Nu este aşa că sînt prea apropiate?

Vă sugerăm să le distanţăm puţin, cu un rînd. Pentru aceasta va trebui să "umblăm" din nou în program şi să introducem tot o instrucţiune **PRINT**, dar, de data aceasta, fără nimic altceva în plus!

Apare o problemă – plasarea lui **PRINT!** Apelaţi în acest caz la comanda care permite revederea liniilor de program, comanda **LIST**.

Editarea unei linii albe: PRINT

Tastaţi **LIST** şi apoi **[RETURN]**. Acum vă este mult mai uşor să localizaţi coordonatele noii instrucţiuni **PRINT** – între liniile 10 şi 20. Realizaţi următoarea succesiune de tastări:

15 **PRINT [RETURN]**
LIST [RETURN]
RUN [RETURN]

Sperăm că aţi obţinut o imagine similară celei de mai jos.

```
INVATAM BASIC...

...CONVERSIND CU CALCULATORUL
```

Nu-i aşa că textul afişat este mult mai lizibil? Totul se datorează instrucţiunii **PRINT** din linia 15. Spunem **PRINT**, dar nu se indică nimic, deci

nu se tipăreşte nimic, astfel încît se introduce automat un spaţiu între cele două linii (10 şi 20). Deci, acesta este modul de introducere a·unei linii cu spaţii.

Să reţinem, în consecinţă, următoarele noi reguli:

Regula 5. **Fiecare instrucţiune PRINT imprimă cîte o linie pe suprafaţa display-ului (televizorului).**

Regula 6. **În cazul în care linia program nu încape pe o linie a display-ului (televizorului) se va continua pe linia următoare.**

Regula 7. **Instrucţiunea PRINT fără nici o opţiune imprimă o linie albă.**

Aplicaţie. Modificaţi programul anterior, astfel încît pe ecranul televizorului să apară, cu spaţierea necesară (v. copertă) şi numărul volumului (1, de exemplu) cît şi numele editurii – EDITURA TEHNICA. Atenţie la alinieri!

Observaţie. Intr-o conversaţie ulterioară vom relua această aplicaţie, solicitîndu-vă şi imaginea color a copertii!

TEST

Modificaţi programul BASIC realizat mai înainte astfel încît el să afişeze următorul text modificat (atenţie la noile alinieri):

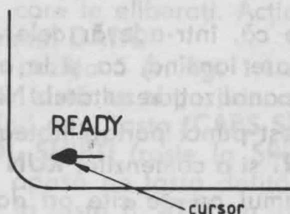
```
                          INVATAM BASIC . . .
         . . . CONVERSIND CU CALCULATORUL aMIC
                                             1
                               EDITURA TEHNICA
```

☐ Imaginea programului BASIC-PRAE

vol. 2, pag. 198

Obţinerea lui READY

Apăsaţi o tastă oarecare, de preferinţă cea cu o săgeată pe ea ⏎ (retur de car) pe care o vom nota în continuare cu **[CR]**. Admiraţi cît doriţi emblema de pe ecran şi apăsaţi, la întîmplare, o tastă. Dacă imaginea obţinută nu conţine mesaje de eroare tastaţi **[CR]**. Calculatorul va afişa numărul de octeţi disponibili în BASIC-PRAE: 7050 octeţi ((7050 BYTES FREE) sau 39818 octeţi (39818 BYTES FREE), în funcţie de capacitatea memoriei calculatorului dvs. (16 Ko RAM respectiv 48 Ko RAM).



Remarcaţi cum pe ecran şi-au făcut deja apariţia **READY** şi cursorul – săgeata din colţul din stînga jos care se "agită" tot timpul.

Introducerea şi corectarea interactivă a unei linii

Este momentul să introduceţi, în aceeaşi manieră, ca la, calculatorul personal aMIC, prima linie din program. Verificaţi cu atenţie ceea ce s-a tipărit pe ecran.

```
10 PRINT "INVATAM BASICCCC
```

În programul nostru s-a strecurat o greşeală. Ţinînd ceva mai mult de o secundă degetul pe tasta **[C]**, ea s-a repetat de mai multe ori. Se impune, aşadar, ştergerea ultimelor trei caractere şi înlocuirea acestora cu trei puncte.

La calculatorul PRAE ştergerea unui caracter din faţa cursorului se realizează cu tasta **[DEL]** (**SHIFT** şi Ø). Spre deosebire însă de aMIC, caracterul nu se şterge de pe ecran, ci este afişat între două semne "\".

Să corectăm împreună linia 10, utilizînd pentru ştergerea ultimelor trei caractere tasta **[DEL]**.

Apăsaţi tasta **[DEL]** şi remarcaţi cum, pentru prima dată, pe ecran a apărut "\C", ceea ce înseamnă că s-a şters litera C (ultima literă). La a doua şi a treia apăsare a aceleiaşi taste au reapărut literele C. Tastaţi punctul. Remarcaţi cum pe ecran a apărut mai întîi semnul "\", avertizîndu-ne că operaţia de ştergere s-a terminat. Tastaţi în continuare, de două ori, punctul, apoi ghilimelele.

În momentul cînd aţi terminat de tastat linia program este bine să înştiinţaţi şi calculatorul de intenţia dvs., prin simpla apăsare a tastei **[CR]**. Vi se pare complicat? Dacă aveţi de operat mai multe corecturi şi imaginea liniei de program este prea "încărcată", tastaţi **[CTRL]** şi **[R]**.

Pentru editarea în condiţii de lizibilitate mărită a titlului complet al lucrării, tastaţi aceleaşi instrucţiuni ca şi la aMIC.

15 **PRINT [CR]**
20 **PRINT** "... CONVERSIND CU CALCULATORUL" **[CR]**

Pentru a vă convinge că, într-adevăr cele două instrucţiuni au fost acceptate, cereţi o confirmare lansînd, ca şi la aMIC comanda **LIST.** Tastaţi în ordine **RUN, [CR]** şi analizaţi rezultatul. Nici o problemă, nu-i aşa?

Remarcaţi pînă în acest punct portabilitatea (de la aMIC la PRAE) a celor două instrucţiuni **PRINT** şi a comenzilor **RUN** şi **LIST.**

Puteţi executa programul ori de cîte ori doriţi. Rezultatul va fi însă mereu acelaşi.

Urmează, în sfîrşit, operaţia de ştergere a ecranului. În BASIC-PRAE acest lucru se poate realiza cu instrucţiunea **CLS.**

Aplicaţie. Simulaţi cu creionul şi hîrtia răspunsul calculatorului PRAE în urma execuţiei următorului program:

```
10 PRINT "INVATAM BASIC..."          [CR]
15 PRINT [CR]
20 PRINT "...CU CALCULATORUL"        [CR]
25 PRINT [CR]
30 PRINT "PERSONAL PRAE"             [CR]
RUN [CR]
INVATAM BASIC...
...CU CALCULATORUL
PERSONAL PRAE
```

□ Imaginea programului BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 198

Utilizarea claviaturii

Puneţi în stare de funcţionare calculatorul personal: HC-85, TIM S, SPECTRUM şi faceţi cunoştinţă cu prima imagine (color sau alb-negru) care apare pe monitor (exemplu: H.C.85 I.C.E. FELIX). Deşi sînteţi nerăbdători să introduceţi cît mai repede programul, vă rugăm să mai aşteptaţi puţin. Motivul este unul singur: tastatura! Spre deosebire de celelalte calculatoare cu care aţi lucrat pînă acum, tastatura lui HC-85, TIM S, SPECTRUM este profesională, iar cunoaşterea ei este o consecinţă a lucrului intens cu calculatorul.

Să începem prin a învăţa s-o utilizăm. Două taste sînt cele mai importante: **[CAPS SHIFT]** şi **[SYMBOL SHIFT]**. Identificaţi-le fără întîrziere pentru o mică demonstraţie. Menţineţi apăsată tasta **[CAPS SHIFT]** şi acţionaţi orice tastă alfabetică. Rezultatul! Toate literele care apar pe ecran sînt scrise cu majuscule (v. poziţia 1 din fig. 1.1-a). Menţineţi apăsată tasta **[SYMBOL SHIFT]** şi acţionaţi, la întîmplare, orice tastă a claviaturii (în afară de **[ENTER]**, **[BREAK SPACE]** sau **[CAPS SHIFT]**). Veţi vedea cum pe ecran apar simbolurile din poziţia 2 (v. figura 1.1-b) sau cele scrise în roşu (v. SPECTRUM) sub fiecare din tastele care au fost acţionate (de exemplu, o săgeată verticală pentru tasta H). Apăsaţi acum pe o tastă alfabetică oarecare, fără a acţiona nici una din tastele **[SHIFT]**. Veţi obţine instrucţiunile scrise în poziţia 3 (v. figura 1.1-c) sau pe cele scrise în alb (v. SPECTRUM) **DIM, IF, NEXT** etc. care permit scrierea programelor BASIC.

Pentru a obţine instrucţiunile scrise în poziţia 4 (v. figura 1.1-d) sau pe cele scrise cu verde (v. SPECTRUM) deasupra tastelor, acţionaţi cele două taste **[SHIFT]**, după care le eliberaţi. Acţionaţi, de exemplu, tasta D şi veţi obţine pe ecran cuvîntul **DATA.**

Cuvintele scrise în poziţia 5 (v. fig. 1.1-e) sau cele scrise cu roşu (v. SPECTRUM) sub fiecare tastă se obţin prin apăsarea simultană a celor două taste **[SHIFT]**, eliberînd apoi tasta **[CAPS SHIFT]** (albă la SPECTRUM), în timp ce tasta **[SYMBOL SHIFT]** (roşie la SPECTRUM) se menţine apăsată, perioadă în care se apasă pe tasta dorită. De exemplu, obţineţi cuvîntul **BRIGHT** pornind de la tasta B, **ATTR** cu tasta L şi **VERIFY** cu tasta R (v. figura 1.1-e, f, g).

Observaţie. Aceasta este valabil pentru toate tastele, în afară de funcţiile **EDIT, ENTER, GRAPHICS, DELETE.**

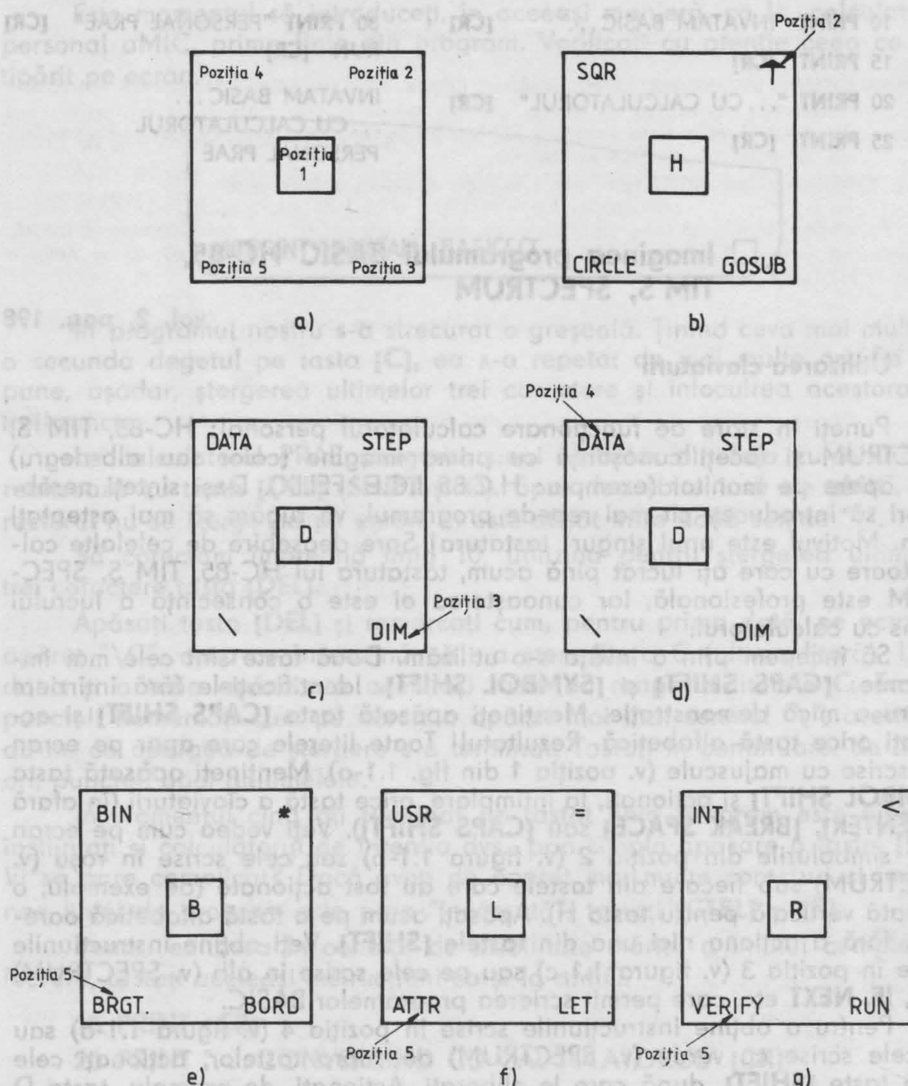


Fig. 1.1. Obținerea funcțiilor și comenzilor cu ajutorul tastelor calculatorului personal HC-85: *a)* pozițiile unei taste a calculatorului personal HC-85; *b)* obținerea simbolurilor din poziția 2 **[SYMBOL SHIFT]** [orice tastă] (v. restricții); *c)* obținerea instrucțiunilor din poziția 3; *d)* obținerea instrucțiunilor din poziția 4: se acționează simultan **[SYMBOL SHIFT]**, **[CAPS SHIFT]**, se eliberează **[SYMBOL SHIFT]** și **[CAPS SHIFT]** și se acționează tasta care conține instrucțiunea dorită; *e, f, g)* obținerea funcțiilor din poziția 5: se acționează simultan **[SYMBOL SHIFT]**, **[CAPS SHIFT]**, se ține apăsată tasta **[SYMBOL SHIFT]** și se acționează tasta care conține funcția dorită.

În figura 1.2 (a, b, c, d, e) sînt prezentate tastele calculatorului personal TIM S corespunzătoare celor din figura 1.1 (b, c, d, e, f, g).

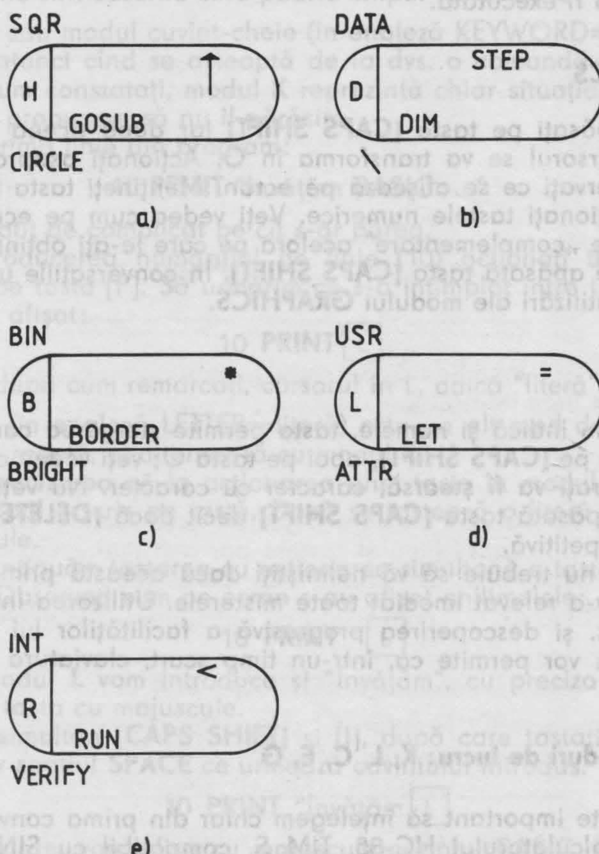


Fig. 1.2. Obţinerea funcţiilor şi comenzilor cu ajutorul tastelor calculatorului personal TIM S.

EDIT

Această tastă este utilizată atunci cînd se doreşte modificarea conţinutului liniilor unui program. Deplasînd cursorul (simbolul ">") şi menţinînd apăsată tasta **[CAPS SHIFT]**, tasta [1/**EDIT**] permite scrierea liniei de editat la baza ecranului; tastele [5] şi [8] permit deplasarea cursorului în lungul liniei pentru eventualele modificări; sensul deplasării este reprezentat prin săgeata situată deasupra fiecăreia din aceste taste.

ENTER

Această tastă serveşte validării liniei de program pe care aţi editat-o; această linie se ataşează astfel în continuarea programului, în par-

tea de sus a ecranului. Dumneavoastră puteţi utiliza această instrucţiune numai după ce ea a fost introdusă în memoria calculatorului, tastînd **RUN** pentru a putea fi executată.

GRAPHICS

Dacă apăsaţi pe tasta **[CAPS SHIFT]** iar după aceea tastaţi 9, veţi vedea cum cursorul se va transforma în G. Acţionaţi asupra tastelor numerice şi observaţi ce se afişează pe ecran. Menţineţi tasta **[CAPS SHIFT]** apăsată şi acţionaţi tastele numerice. Veţi vedea cum pe ecran apar simbolurile grafice „complementare" acelora pe care le-aţi obţinut mai înainte fără a menţine apăsată tasta **[CAPS SHIFT]**. În conversaţiile următoare vom vedea şi alte utilizări ale modului **GRAPHICS.**

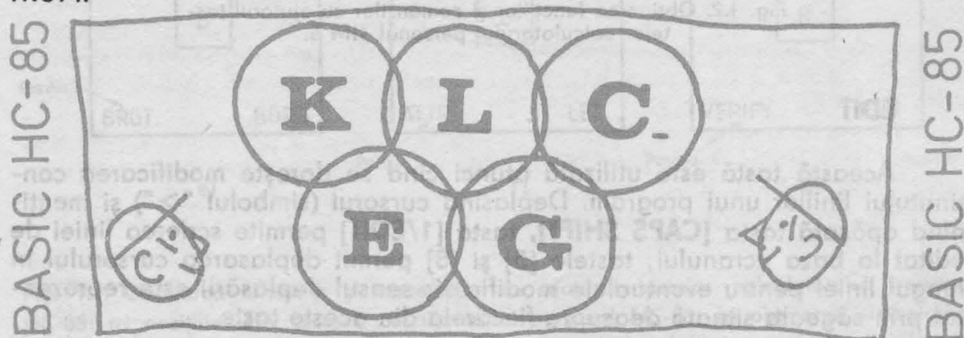
DELETE

După cum indică şi numele, tasta permite ştergerea caracterelor nedorite. Apăsaţi pe **[CAPS SHIFT]**, apoi pe tasta O; veţi vedea cum linia asupra căreia lucraţi va fi ştearsă, caracter cu caracter. Nu veţi avea nevoie să menţineţi apăsată tasta **[CAPS SHIFT]** decît dacă **[DELETE]** va trebui să devină auto-repetitivă.

Desigur, nu trebuie să vă neliniştiţi dacă această primă descriere a claviaturii nu v-a relevat imediat toate misterele. Utilizarea intensivă a calculatorului dvs. şi descoperirea progresivă a facilităţilor lui în funcţie de nevoile dvs. vă vor permite ca, într-un timp scurt, claviatura să vă devină familiară.

Cinci moduri de lucru: K, L, C, E, G

Este foarte important să înţelegem chiar din prima conversaţie modul de lucru al calculatorului HC-85, TIM S, compatibil cu SINCLAIR SPECTRUM.



Aţi dat bineţe simbolului "K" din colţul din stînga jos al ecranului dvs.?

El este cursorul şi vă rugăm, să-i acordaţi toată atenţia. Să nu credeţi că el va avea tot timpul "înfăţişarea" lui **K**. Nu! Deseori se schimbă în **L**, apoi în **C** şi nu puţine sînt cazurile cînd poartă chipul lui **E** sau **G**.

Modul K sau modul cuvînt-cheie (în engleză KEYWORD=cuvînt cheie) apare numai atunci cînd se aşteaptă de la dvs. o comandă sau linie program. După cum constataţi, modul **K** reprezintă chiar situaţia noastră şi, în consecinţă, vă propunem să nu îl părăsim.

Tastaţi prima linie din program:

10 **PRINT** "Învăţăm BASIC . . ."

Nu este chiar atît de complicat pe cît s-ar părea.

Pentru introducerea numărului de linie (10) acţionaţi tastele 1 şi 0. Apăsaţi apoi pe tasta [P]. Să urmărim ce s-a întîmplat între timp pe ecran! Calculatorul a afişat:

10 **PRINT** [L]

transformînd, după cum remarcaţi, cursorul în L, adică "literă".

Modul L (în engleză LETTER=literă) este un alt mod de lucru al calculatorului; de regulă, el alternează cu modul **K**.

Facem precizarea că la acţionarea unei taste în modul **L** se afişează simbolul principal înscris pe tastă. Dacă se tastează o literă ea va fi tipărită cu minuscule.

Să ne continuăm tastarea cu acţionarea simultană a tastelor **[SYMBOL SHIFT]** şi [P]. Observaţi cum pe ecran s-au afişat ghilimelele:

10 **PRINT** "[L]

Tot în modul **L** vom introduce şi "Învăţăm", cu precizarea că primul caracter se va tasta cu majuscule.

Apăsaţi simultan **[CAPS SHIFT]** şi [I], după care tastaţi restul caracterelor, inclusiv spaţiul **SPACE** ce urmează cuvîntului introdus.

10 **PRINT** "Învăţăm [L]

În continuare, vom afişa cu majuscule cuvîntul BASIC. Modul în care se realizează scrierea cu litere mari este **modul C** (în engleză CAPITALS=capitale), o variantă a modului **L**. Trebuie deci schimbată înfăţişarea cursorului din **L** în **C**. Tastaţi simultan **[CAPS SHIFT]** şi [2]. Acţionaţi în continuare asupra tastelor care conţin caracterele B, A, S, I şi C. Remarcaţi cum, din acest moment, ne aflăm cu cursorul în modul **C**.

10 **PRINT** "Învăţăm BASIC [C]

Urmează să introduceţi cele trei puncte prin apăsarea simultană a tastelor **[SYMBOL SHIFT]** şi [M]. Nu uitaţi ghilimelele! (**[SYMBOL SHIFT]**, [P]).

10 **PRINT** "Învăţăm BASIC . . ." [C]

Înainte de a apăsa pe **[ENTER]** verificaţi cu atenţie corectitudinea celor introduse. Dacă aveţi "neplăceri" folosiţi tasta **[DELETE]** (**[CAPS SHIFT]**, [0]). Ea permite ştergerea unui caracter din stînga cursorului.

După ce aţi eliminat şi eventualele caractere eronate ("scăpate" la tastare), apăsaţi pe **[ENTER]** şi remarcaţi cum între numărul de linie (10) şi **PRINT** s-a intercalat simbolul ">" pe care îl vom numi cursorul programului.

```
10 >PRINT "Invăţăm BASIC..."




C
```

Simbolul ">" marchează, de regulă, linia curentă. Este vorba în principal de ultima linie introdusă.

Observaţie. Pentru urcarea şi coborîrea simbolului ">" puteţi utiliza tastele [↑] respectiv [↓].

Nu-i aşa că dialogul pe care îl purtaţi acum cu HC-85, TIM S, SPECTRUM a devenit incitant? Pînă şi tastele par a fi mai prietenoase. Vă propunem să introducem în continuare cea de-a doua linie a programului:

20 **PRINT** ". . . conversînd cu calculatorul" **[ENTER]**

Tastaţi:

R [ENTER]

şi priviţi cum pe ecranul monitorului s-au afişat, fără spaţiu între ele, cele două rînduri care alcătuiesc titlul lucrării.

```
Invăţăm BASIC...
... conversînd cu calculatorul




K
```

Pentru tipărirea titlului cărţii în condiţii de lizibilitate mărită, tastaţi în continuare următoarele comenzi şi instrucţiuni BASIC:

LIST	**[ENTER]**	15 **PRINT**	**[ENTER]**
5 **CLS**	**[ENTER]**	**LIST**	**[ENTER]**
LIST	**[ENTER]**	**RUN**	**[ENTER]**

Observaţie. **Modul E** (în engleză EXTENDED=extins) este utilizat pentru a obţine caractere noi, în special alte comenzi. **Modul G** (în engleză GRAPHICS = grafic) este folosit în programarea grafică. Atît modul **E** cît şi modul **G** vor face obiectul unor conversaţii viitoare.

□ Imaginea programului BASIC-AMSTRAD

vol. 2, pag. 198

Introducere în BASIC-AMSTRAD

Remarcaţi cum, odată cu "trezirea" AMSTRAD-ului, pe ecranul monitorului dvs. se afişează cu repeziciune următorul text:

```
Amstrad 128 K Microcomputer(V3)
© 1985 Amstrad Consumer Electronics plc
  and Locomotive Software Ltd.

Basic 1.1
Ready
```



Ceea ce surprinde în mod plăcut la acest calculator este dorinţa lui imediată de conversaţie, fără nici un alt protocol. El este gata **(READY)** de a primi în ospeţie programul dvs. Pentru a vă obişnui cu noua gazdă începeţi prin a introduce prima linie:

10 **print** "Învăţăm BASIC . . ."

Aţi obţinut şi dvs. această imagine?

```
10 print " Învăţăm BASIC..." ■
```

Dacă nu, înseamnă că la tastare s-au strecurat greşeli. Vă puteţi folosi în acest caz de tasta **[DEL]** care permite ştergerea oricărui caracter din stînga cursorului. Nu este nimic complicat!

Să mai rămînem totuşi la această imagine şi pentru motivul unor explicaţii suplimentare, dacă consideraţi că ele vă sînt utile.

Ce vi se pare mai nou, atît la tipărirea instrucţiunii **PRINT** cît şi la tipărirea anumitor caractere din cadrul mesajului?

Deoarece atît litera I de la începutul mesajului cît şi literele ce alcătuiesc cuvîntul BASIC au fost tastate cu **[SHIFT]**, ele au fost reproduse, evident, cu majuscule. Cît priveşte instrucţiunea **PRINT**, ea a fost tipărită cu minuscule, întrucît la tastare nu am "forţat" **[SHIFT]**-ul.

Observație. Pentru introducerea unei instrucțiuni în BASIC-AMSTRAD puteți utiliza indiferent ce caractere doriți: MAJUSCULE sau minuscule.

Dacă sînteți mulțumiți de modul în care ați introdus prima linie, tastați **[RETURN]** sau **[ENTER]** și apoi executați programul cu:

run **[RETURN]**

Veți vedea cum pe ecran a apărut titlul lucrării:

```
Invăţăm BASIC...
Ready
■
```

Observație. Dacă doriți, puteți înlocui **PRINT** cu semnul întrebării:

10 **?** "Învățăm BASIC . . ." **[RETURN]**

Cum vi se pare AMSTRAD-ul? Vă place să conversați împreună? Ne-ar surprinde un răspuns negativ. De aceea, vă și invităm să tastați în continuare următoarea succesiune de instrucțiuni și comenzi:

15 **?** **[RETURN]**
20 **?** „. . . conversînd cu calculatorul" **[RETURN]**
list **[RETURN]**
run **[ENTER]**

Dacă ați urmărit atent dialogul cu calculatorul AMSTRAD, ați remarcat cum la comanda **list** transmisă de dvs., el a răspuns prin a tipări cu majuscule instrucțiunile **PRINT** pe care dvs. le-ați introdus cu semnul întrebării.

Pentru ștergerea ecranului, introduceți instrucțiunea:

5 **cls** **[RETURN]**

În continuare se prezintă "textul" conversației cu calculatorul AMSTRAD; remarcați cum, la ultima comandă **RUN,** pe ecranul monitorului șters de instrucțiunea **CLS,** s-a afișat ceea ce, de altfel, ne-am propus: titlul cărții!

Ready
10 **?** "Învățăm BASIC . . ."
run
Învățăm BASIC
Ready
15 **?**
20 **?** ". . . conversînd cu calculatorul"
list
10 **PRINT** "Învățăm BASIC . . ."
15 **PRINT**
20 **PRINT** ". . . conversînd cu calculatorul"
Ready
run
Învățăm BASIC . . .
. . . conversînd cu calculatorul
Ready
5 **cls**
list
5 **CLS**
10 **PRINT** "Învățăm BASIC . . .
15 **PRINT**
20 **PRINT** ". . . conversînd cu calculatorul"
Ready
run
Învățăm BASIC . . .
. . . conversînd cu calculatorul
Ready
■

Trei moduri de afişaj

Calculatorul personal AMSTRAD dispune de trei moduri de afişaj: Mode Ø, Mode 1 şi Mode 2. În mod automat, calculatorul lucrează în Mode 1. Pentru a înţelege cele trei moduri de afişaj, vă propunem o mică demonstraţie în trei paşi.

· **Pasul 1.** Apăsaţi pe orice tastă, de exemplu [1], pînă cînd obţineţi două linii cu 1. Număraţi apariţiile cifrei 1 pe una din linii. Aţi găsit 40? Tastaţi în continuare **[RETURN]** sau **[ENTER]**. Veţi obţine un mesaj de eroare ⟨⟨Syntax Error⟩⟩. Nu vă alarmaţi! Am ales cel mai scurt drum pentru a reveni la **READY**.

Pasul 2. Tastaţi:

mode Ø [RETURN]

Veţi observa cum pe ecranul monitorului se obţin caractere mult mai mari. Apăsaţi din nou pe tasta [1] pînă obţineţi două linii pline. Vă rugăm să le număraţi. Aţi găsit 20 ? Tastaţi **[RETURN]**.

Pasul 3. Tastaţi:

mode 2 [RETURN]

Apăsaţi pe tasta [1] pînă obţineţi două linii pline. Dacă număraţi cu atenţie apariţiile acestei cifre într-una din linii veţi găsi 80. Remarcaţi cum scrisul (corpul de literă), în comparaţie cu cazurile precedente, apare mult mai mic.

În concluzie:

Mode Ø=20 coloane
Mode 1=40 coloane
Mode 2=80 coloane

Observaţie. Pentru a pune calculatorul la zero **(Mode 1)** apăsaţi pe **[CONTROL] [SHIFT]** şi **[ESC]** în această ordine.

Aplicaţie. Executaţi programul BASIC creat anterior utilizînd toate cele trei moduri de afişaj, specifice calculatorului AMSTRAD.

a) Modul zero

```
mode Ø [RETURN]                      20 ? "... conversînd cu calculatorul"
 5 cls [RETURN]                         [RETURN]
10 ? "Învăţăm BASIC..." [RETURN]     run [RETURN]
15 ? [RETURN]                           [CONTROL], [SHIFT], [ESC]
```

b) Modul unu

```
mode 1 [RETURN]                      20 ? "... conversînd cu calculatorul"
 5 cls [RETURN]                         [RETURN]
10 ? "Învăţăm BASIC..." [RETURN]     run [RETURN]
15 ? [RETURN]                           [CONTROL], [SHIFT], [ESC]
```

c) Modul doi

```
mode 2 [RETURN]                      20 ? "... conversînd cu calculatorul"
 5 cls [RETURN]                         [RETURN]
10 ? "Învăţăm BASIC..." [RETURN]     run [RETURN]
15 ? [RETURN]                           [CONTROL], [SHIFT], [ESC]
```

În BASIC-AMSTRAD cele trei moduri de afişaj pot fi obţinute cu instrucţiunea **MODE,** similară comenzii **MODE,** cu singura deosebire că in-

strucţiunea **MODE,** ca de altfel orice instrucţiune BASIC, trebuie să fie precedată de un număr de linie.

În cele ce urmează, vă propunem să testaţi şi facilităţile acestei instrucţiuni, foarte des utilizate în programarea pe calculatorul AMSTRAD, după cum urmează:

3 mode Ø [RETURN]	**3 mode 1 [RETURN]**
5 cls [RETURN]	**list [RETURN]**
10 ? "Învăţăm BASIC . . ." **[RETURN]**	**run [RETURN]**
15 ? [RETURN]	**3 mode 2 [RETURN]**
20 ? ". . . conversînd cu calculatorul" **[RETURN]**	**list [RETURN]**
run [RETURN]	**run [RETURN]**

Observaţie. Pentru a vă reîntoarce în **MODE 1** nu uitaţi să tastaţi **[CONTROL], [SHIFT], [ESC]** în această ordine.

În cele ce urmează se prezintă textul îmbunătăţit al conversaţiei purtate cu calculatorul AMSTRAD.

Ready	10 **?** "Învăţăm BASIC . . ."
3 mode Ø	15 **?**
5 cls	20 **PRINT** ". . . conversînd cu calculatorul"
	run

```
Invatam BASIC...
...conversind cu calculatorul
```

Ready	10 **PRINT** "Învăţăm BASIC . . ."
3 **mode 1**	15 **PRINT**
list	20 **PRINT** ". . . conversînd cu calculatorul"
3 **MODE 1**	**Ready**
5 **CLS**	**run**

```
Invăţăm BASIC...
...conversînd cu calculatorul
```

Ready	10 **PRINT** "Învăţăm BASIC . . ."
3 **mode 2**	15 **PRINT**
list	20 **PRINT** ". . . conversînd cu calculatorul"
3 **MODE 2**	**Ready**
5 **CLS**	**run**

Invăţăm BASIC...
...conversînd cu calculatorul

☐ Imaginea programului BASIC-COMMODORE

vol. 2, pag. 198

Accesul la calculator

Odată cu "trezirea" calculatorului COMMODORE, pe ecranul monitorului se afişează mesajul:

```
***COMMODORE 64 BASIC V2***
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY
```



Sînteţi pregătit pentru introducerea programului? Cunoaşteţi bine tastatura calculatorului COMMODORE? Să începem prin a tasta prima linie:

10 **PRINT** "ÎNVĂŢĂM BASIC . . ." **[RETURN]**

De notat că **PRINT**, cea mai utilizată instrucţiune în BASIC poate fi înlocuită cu semnul întrebării (?).

Tastaţi instrucţiunile şi comenzile următoare, apăsînd de fiecare dată pe tasta **[RETURN]**.

RUN [RETURN]
15 **PRINT [RETURN]**
20 **PRINT** ". . . CONVERSÎND CU CALCULATORUL" **[RETURN]**
LIST [RETURN]

RUN [RETURN]
5 **PRINT CHR$ (147) [RETURN]**
RUN [RETURN]

Instrucţiunea **PRINT CHR$** (147) din linia 5 serveşte la ştergerea ecranului, dar nu şi a programului. Pentru ştergerea unui caracter eronat dintr-o linie, veniţi cu cursorul pe locul dorit (**[SHIFT]**, **[CRSR]**), şi acţionaţi **[SHIFT]**, **[INST]**.

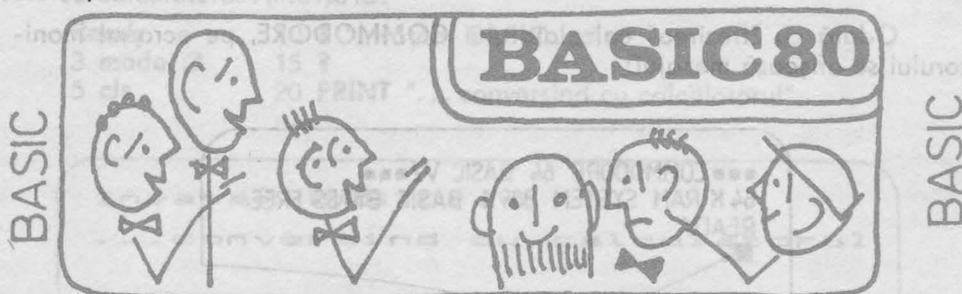
□ Imaginea programului BASIC-80*, GW-BASIC pentru Felix PC

vol. 2, pag. 199

Operare, comenzi și instrucțiuni

Aveți în fața dvs. un microcalculator românesc: M118, JUNIOR, TPD. Înțelegem emoția conversației cu aceste calculatoare dar nu aveți de ce să vă fie... teamă. V-ați acomodat cu tastatura? Ați identificat [SHIFT]-ul, [RETURN]-ul și celelalte taste mai importante? Aveți definiți toți pașii pe care trebuie să-i parcurgeți singuri? Dacă nu, să-i listăm împreună: 1) încărcarea interpretorului BASIC-80; 2) obținerea lui **READY**; 3) introducerea primei linii a programului; 4) ștergerea (dacă este cazul) a caracterelor unei linii program; 5) execuția programului; 6) extinderea programului; 7) listarea programului; 8) ștergerea ecranului.

Și-acum, la muncă!

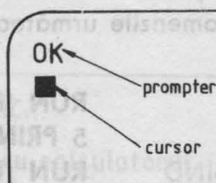


Introduceți în unitatea de floppi discheta care conține interpretorul BASIC-80. Tastați comanda CP/M:

A>BASIC 80 [RETURN]

și așteptați apariția pe ecranul monitorului a mesajului „OK" (omologul lui **READY**). Nu ignorați prezența acestuia! Din acest moment puteți introduce prima linie program:

10 **PRINT** "Învățăm BASIC..." **[RETURN]**



Dacă în momentul tastării liniei program ați remarcat apariția unor caractere eronate, există posibilitatea corectării lor (v. conversația 3, comanda **EDIT**). Pentru introducerea acestei linii în memoria calculatorului tastați **[RETURN]**.

* BASIC-80 asemănător cu GW-BASIC, de pe Felix PC (IBM PC). Acesta este tratat în detaliu în vol. 2, pag. 66. Observația este comună programelor BASIC-80 din întreaga lucrare.

Este bine să reţineţi că în BASIC-80 numărul liniei este cuprins între 0 şi 65529 iar o linie program poate avea maximum 255 caractere. Cît priveşte instrucţiunea **END**, prezenţa acesteia nu este obligatorie.

Executaţi programul cu comanda:

RUN [RETURN]

şi analizaţi primul rezultat.

În vederea extinderii programului cu cele două instrucţiuni **PRINT**, tastaţi în continuare

15 **PRINT [RETURN]**
20 **PRINT** "...conversînd cu calculatorul" **[RETURN]**

Listaţi programul cu comanda

LIST [RETURN]

şi introduceţi instrucţiunea

5 **PRINT CHR$ (24) [RETURN]**

pentru ştergerea ecranului. Nu uitaţi denumirea "buretului de ştergere" în BASIC-80 – **CHR$ (24)** ce se ataşează unei instrucţiuni **PRINT**.

În continuare, puteţi urmări întreaga conversaţie purtată în BASIC-80 cu microcalculatoarele româneşti M118, JUNIOR, TPD.

```
OK
10 PRINT "Învăţăm BASIC..."
RUN
Învăţăm BASIC...
OK
15 PRINT
20 PRINT "...conversînd
          cu calculatorul"
LIST
10 PRINT "Învăţăm BASIC..."
15 PRINT
20 PRINT "...conversînd
          cu calculatorul"
OK
RUN
Învăţăm BASIC...
...conversînd cu calculatorul
OK
 5 PRINT CHR$ (24)
LIST
 5 PRINT CHR$ (24)
10 PRINT "Învăţăm BASIC..."
15 PRINT
20 PRINT "...conversînd
          cu calculatorul"
OK
RUN
Învăţăm BASIC...
...conversînd cu calculatorul
OK
```

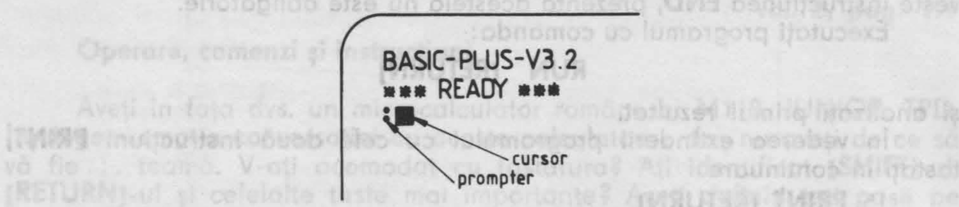
☐ Imaginea programului BASIC-PLUS, pentru INDEPENDENT, CORAL

vol. 2, pag. 199

READY, prompter, cursor

Pentru a pune "la treabă" calculatoarele INDEPENDENT şi CORAL trebuie să obţineţi mai întîi: *** **READY** *** şi simbolul „ : " pe care îl vom numi **prompter. Prompterul** reprezintă modul de exprimare al celor două calculatoare: "mergeţi înainte, faceţi ceva!" (în engleză, to prompt=a împinge, a îndemna). Spre deosebire de mesajul **READY,** prompterul apare de fiecare dată cînd sistemul aşteaptă o nouă linie de program, chiar dacă

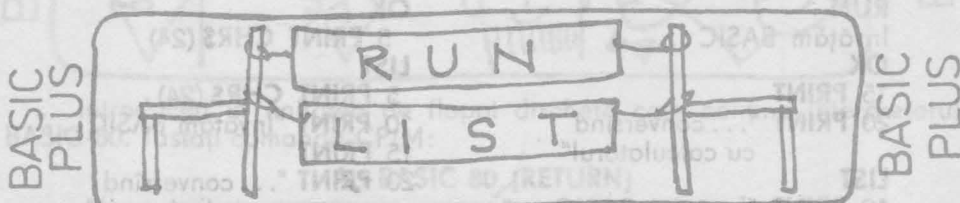
linia respectivă se întîmplă să fie prima sau nu. Remarcaţi cum, sub forma unei linii subţiri, şi-a făcut apariţia şi cursorul.



Deci, în rezumat, cînd calculatoarele INDEPENDENT şi CORAL afişează pe ecran **READY**, ele spun de fapt: "am înregistrat comanda şi sîntem gata să acceptăm o nouă linie de program". Simbolul „ : " denumit **prompter** se identifică cu "sînt pregătit să accept o nouă linie de program", iar simbolul cursor are semnificaţia ştiută: "aici este locul unde următorul caracter va apare pe ecran".

Singuri printre ... comenzi şi instrucţiuni

Cum şi calculatoarele INDEPENDENT şi CORAL se "hrănesc" tot cu programe, considerăm oportună introducerea de la tastatură a programului nostru. De această dată vă vom lăsa singuri, invitîndu-vă să tastaţi următoarea succesiune de instrucţiuni şi comenzi:



10 **PRINT** "ÎNVĂŢĂM BASIC. . ." **[RETURN]**
99 **END** **[RETURN]**
RUN **[RETURN]**
15 **PRINT** **[RETURN]**
20 **PRINT** ". . . CONVERSÎND CU CALCULATORUL" **[RETURN]**
LIST **[RETURN]**
RUN **[RETURN]**

Prezentăm în continuare întreaga conversaţie:

: 10 **PRINT** "ÎNVĂŢĂM BASIC . . ."
: 99 **END**
: **RUN**
ÎNVĂŢĂM BASIC . . .
END OF PROGRAM AT LINE 99
: 15 **PRINT**
: 20 **PRINT** ". . . CONVERSÎND
CU CALCULATORUL"
: **LIST**

10 **PRINT** "ÎNVĂŢĂM BASIC . . ."
15 **PRINT**
20 **PRINT** ". . . CONVERSÎND CU
CALCULATORUL"
99 **END**
: **RUN**
ÎNVĂŢĂM BASIC . . .
. . . CONVERSÎND CU
CALCULATORUL
END OF PROGRAM AT LINE 99

Cum vi s-a părut începutul acestei . . . aventuri? Chiar dacă nu aţi avut "probleme", cîteva precizări sînt totuşi necesare:

Reguli

- Ştergerea caracterelor unei linii program se realizează prin acţionarea tastei **[DELETE]** (RUB OUT). Fiecare acţionare determină ştergerea unui caracter.
- Semnalul de terminare a unei linii program constă în apăsarea tastei **[RETURN]**.
- Listarea unui program (sau a unei secţiuni din el) se realizează cu comanda **LIS[T]** (T este opţional).
- Execuţia unui program existent în memoria internă se realizează cu comanda **RUN**.
- O linie BASIC se poate continua pe mai multe linii terminal. Marcarea continuării liniei program curente pe linia terminal următoare se indică prin simbolul "&" **[RETURN]** plasat în penultima poziţie a liniei care se continuă.

□ Imaginea programului ABASIC, pentru FELIX C

vol. 2, pag. 199

Comanda ABASIC

Interpretorul BASIC sub ARIEL, cunoscut sub numele de ABASIC poate fi lansat de către un utilizator ARIEL (aflat în stare EDITOR) prin comanda:

A [BASIC] [/⟨nume fişier⟩/[,⟨nr. linie⟩]]



unde, ⟨nume fişier⟩ (dacă este prezent) reprezintă numele fişierului în care se găseşte programul BASIC, iar ⟨nr. linie⟩ (dacă este prezent) indică linia din fişierul de intrare – program de la care se începe execuţia programului.

Linia ABASIC

Linia ABASIC conţine maximum 80 de caractere, provenite din articolele unui fişier de intrare sau de la terminal. Numărul de linie precede instrucţiunile, indiferent dacă linia ABASIC provine dintr-un fişier sau de la terminal. Numărul de linie ABASIC poate lua valori cuprinse între 1 şi 32 000. Numim program ABASIC secvenţa de linii ABASIC terminată printr-o instrucţiune **END** sau prin sfîrşit de fişier de intrare-program, ordonată în mod crescător după numărul de linie, indiferent de ordinea în care au fost introduse liniile.

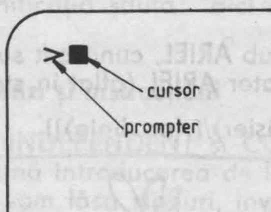
Moduri de lucru în ABASIC

Există trei moduri de lucru în ABASIC: 1) **execuţie imediată** (ABASIC aşteaptă sau prelucrează o instrucţiune imediată sau o comandă de la terminal); 2) **execuţie program** (ABASIC execută un program încărcat, din-

tr-un fişier de intrare sau terminal, ca urmare a unei comenzi sau instrucţiuni); 3) **schimb de date cu terminalul** (programul scrie/citeşte date la/de la terminal). Trecerea de la un mod de lucru la altul se face prin: comanda **A [BASIC]**, instrucţiunile **STOP, END,** apariţia unor erori netratate, **BREAK,** comenzile **RUN** şi **CON,** orice instrucţiune de scriere/citire etc.

Introducerea şi execuţia unui program ABASIC

Linia program ABASIC poate fi tastată la terminal numai după apariţia prompterului ">". Pentru ca linia să fie introdusă în memoria calculatorului, se va acţiona tasta **[RETURN].** Pentru listarea şi execuţia programului folosiţi comenzile **LIST** respectiv **RUN.** În continuare, se prezintă imaginea programului ABASIC executat pe calculatorul FELIX C-512.



Observaţii

- În ABASIC instrucţiunea **PRINT** poate fi înlocuită cu semnul întrebării (**?**).
- Pentru ştergerea ecranului utilizaţi instrucţiunea **CLEAR$** sau **CHR$** (H1b).

```
>10 ? "ÎNVĂŢĂM BASIC..."
>90 END
> RUN
ÎNVĂŢĂM BASIC...
STOPPED AT LINIE 90
>
>15 ?
>20 ?"...CONVERSÎND
    CU CALCULATORUL"
> LIST
```

```
10 PRINT "ÎNVĂŢĂM BASIC..."
15 PRINT
20 PRINT "...CONVERSÎND
    CU CALCULATORUL"
90 END
>
> RUN
ÎNVĂŢĂM BASIC...
...CONVERSÎND CU
   CALCULATORUL
STOPPED AT LINE 90
>
```

TEMA 1

□ Întrebări de control din text

Răspundeţi, ajutîndu-vă de textul nostru, la următoarele întrebări:

1) Există numeroase posibilităţi de comunicare între om şi calculator. Precizaţi care din următoarele variante permit introducerea unui program în calculator: a) ecranul de televizor; b) claviatura; c) joystick.

2) Numiţi cel puţin două funcţiuni îndeplinite de numărul de linie al unui program BASIC.

3) Definiţi următorii termeni:

a) prompter; b) cursor; c) şir de caractere.

4) Care este semnificaţia apariţiei mesajului **READY** pe monitor? Care este diferenţa dintre **READY** şi simbolul cursor/prompter?

5) Care este diferenţa dintre o comandă şi o instrucţiune BASIC?

6) Descrieţi modul în care este posibil să corectaţi o instrucţiune/comandă înainte de a fi apăsat pe **[RETURN]**.

7) Descrieţi procedura pentru ştergerea completă a unei linii din program.

8) Descrieţi procedura pentru inserarea unei noi linii de program într-un program BASIC existent.

9) Descrieţi procedura pentru schimbarea unei linii de program existente, după ce a fost deja introdusă în memoria calculatorului.

10) Se consideră următorul program BASIC:

```
1 PRINT "A"          180 PRINT "A+B"
10 PRINT "B"         9999 END
```

După cum observaţi, în acest exemplu numărul de linie poate lua valori de la 1 la 9 999. Limita superioară diferă de la calculator la calculator. Precizaţi această valoare atît pentru calculatorul dvs. cît şi pentru calculatoarele CORAL, FELIX C, HC-85, AMSTRAD, aMIC şi PRAE.

11) Numerele din faţa liniilor nu sînt lipsite de importanţă. Ele precizează calculatorului ordinea în care trebuie folosite instrucţiunile din program. Nu este necesar ca aceste numere să se succeadă în secvenţa 1, 2, 3, 4, . . . Este mult mai bine şi chiar se recomandă ca numerotarea liniilor să se facă din 10 în 10 aşa cum de altfel am procedat şi noi. Aceasta este o metodă comună de secvenţare. Apoi, dacă dorim adăugarea unor noi linii, ele pot fi mult mai uşor introduse printre cele deja existente. Precizaţi cîte linii pot fi introduse între liniile 30 şi 40 ale următorului program.

```
10 PRINT "LINIA 10"      40 PRINT "LINIA 40"
20 PRINT "LINIA 20"      50 PRINT "LINIA 50"
30 PRINT "LINIA 30"      99 END
```

12)

```
20 PRINT                          5 CLS
80 PRINT "DIMINETI CU FERESTRE    10 PRINT "DIMINEATA DE SEPTEMBRIE"
   DESCHISE"
```

Acesta este un program simplu ce cuprinde .... linii. Fiecare linie începe cu .... Încercuiţi numărul de linie al instrucţiunii pe care calculatorul o va executa prima dată. Precizaţi funcţiunea acestei instrucţiuni.

13) Dacă vă aflaţi în faţa calculatorului personal PRAE imediat după "trezirea" acestuia şi doriţi să introduceţi acest scurt program:

```
5 CLS                    20 PRINT
10 PRINT "TEST BASIC"    30 PRINT "BASIC PRAE"
```

procedaţi astfel: .... , ... , ... , ... ,
.... , ... , ... , ... , ...

Să presupunem că aţi introdus programul şi în continuare doriţi să-l şi executaţi. Tastaţi comanda .... şi apăsaţi pe tasta .... Dacă nu

aţi făcut erori de scriere la introducerea programului, calculatorul va afişa .... Precizaţi textul conversaţiei cu calculatorul PRAE începînd cu introducerea primei instrucţiuni şi terminînd cu afişarea lui **READY**. Aceeaşi întrebare şi pentru calculatoarele HC-85 şi AMSTRAD.

....

....

....

14) Dacă doriţi să eliminaţi o linie dintr-un program fără a o înlocui cu alta, tastaţi .... şi apăsaţi pe .... Următorul program se află în memoria calculatorului:

```
10 CLS              30 ?
20 ? "4+8"          40 ? "4 * 8"
                    50 ?
```

Dorim să eliminăm linia 20. Tastaţi .... şi apăsaţi pe .... Pentru a vă convinge de acest lucru, cereţi o confirmare din partea calculatoarelor .... cu comanda .... **[RETURN]**. Ce se va afişa pe ecranul monitorului?

15) Precizaţi rezultatul execuţiei următoarelor programe:

a)
```
10 PRINT "A"
20 PRINT
30 CLS
40 PRINT "B"
50 PRINT "D"
60 END
```

b)
```
10 PRINT "A"
20 PRINT
30 PRINT "B"
40 CLS
50 PRINT "D"
60 END
```

c)
```
10 PRINT "A"
20 PRINT
30 PRINT "B"
40 PRINT "D"
50 CLS
60 END
```

16) Scrieţi un program BASIC care să editeze:

																						2							
A	R	I	A		C	E	R	C	U	L	U	I		=		3	.	1	4	*	R								

şi

									1										2										3
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0

17) Cînd scrieţi un program BASIC se pot strecura şi greşeli de tastare, ca de exemplu:

```
10 PLINT "A–B"
```

S-a greşit la tastarea lui **PRINT**! Calculatoarele ...., ..., semnalează imediat eroarea de sintaxă (în momentul introducerii liniei), pe cînd calculatoarele ...., ..., ..., nu ne spun că am greşit decît în momentul execuţiei programului. Semnalarea erorilor de către calculator se poate face în mai multe moduri, dar nu aceasta este important, ci faptul că trebuie să ne dăm seama că am bătut L în loc de R şi să corectăm greşeala ajutîndu-ne de .... Nu uitaţi să listaţi după aceea programul.

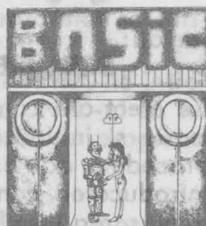
18) Priviţi acest program (alcătuit dintr-o singură linie).

```
10 PRINT "Învăţăm PASCAL. . ."
```

Precizaţi în urma cărei comenzi **(LIST, RUN)** dispar ghilimelele?

CONVERSAŢIA 2

Calculul ariei unui rezervor sferic de rază dată. Constante şi variabile numerice. Expresii numerice. Instrucţiunile LET şi PRINT. Comenzi BASIC: SCR, NEW. TESTE şi APLICAŢII pentru cititor



EXEMPLELE 2 – PC, m, M, F

□ Cîteva operaţii fundamentale ale calculatoarelor şi descrierea lor în BASIC-aMIC

vol. 2, pag. 199

Operaţii numerice

Poate să realizeze aMIC-ul operaţii matematice? Da, poate! Aritmetica elementară este uşoară pentru el. Poate efectua chiar calcule matematice complicate dar şi efortul de programare este mai mare.

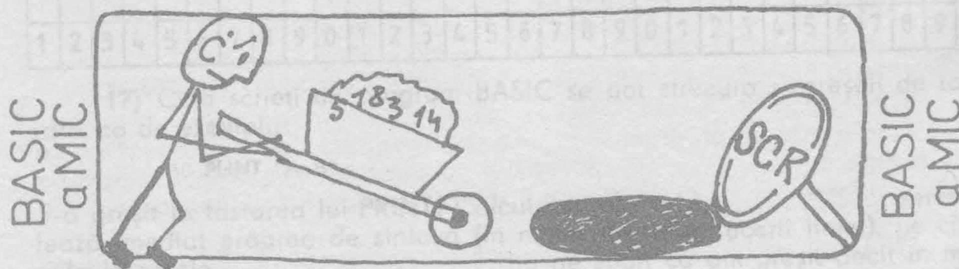
BASIC-ul foloseşte cele patru operaţii matematice fundamentale: adunarea (+); scăderea (–); înmulţirea (*); împărţirea (/) şi include, de asemenea, un număr de funcţii exponenţiale, trigonometrice etc.

Semnul "=" este şi el necesar. BASIC-ul este puţin particular în ceea ce priveşte utilizarea acestui semn. O expresie aritmetică de genul 5*8–3 poate fi efectuată numai dacă se află în partea dreaptă a semnului de egalitate. Partea stîngă a semnului egal este pentru **„nume variabilă"**. Acesta este numele pe care îl dăm rezultatului expresiei aritmetice. Pare ciudat ceea ce v-am spus, dar, după cum veţi vedea peste cîteva pagini, este foarte simplu.

Observaţie. Nu putem folosi "X" pentru operaţia de înmulţire. Din nefericire, folosirea lui "X", care este de fapt o literă, pentru înmulţire, s-a stabilit mult mai demult. Folosim litere pentru alte scopuri, iar utilizarea lui "*" este mult mai puţin generatoare de confuzii pe care calculatorul nu le tolerează.

Comanda SCR

Iniţial, trebuie să eliminăm programul existent în memoria calculatorului. Cel mai drastic mod de distrugere este executarea comenzii **SCR.**



Tastaţi:

SCR [RETURN]

Ce efect are comanda **SCR? SCR** şterge toate programele existente în memorie şi iniţializează toate variabilele cu zero. Este puţin cam dur acest procedeu, dar este o cale bună pentru a porni din nou totul de la zero. Introducerea comenzii **SCR** mai rezolvă şi unele "treburi casnice" cum ar fi plasarea lui **READY** şi a cursorului în colţul din stînga sus.

Tastaţi:

LIST [RETURN]

pentru o confirmare din partea calculatorului a faptului că în memorie nu mai există nici un program.

Program BASIC pentru calculul ariei unui rezervor sferic

Vom folosi în continuare calculatorul pentru rezolvarea unei probleme simple: calculul ariei unui rezervor sferic. Vă mai reamintiţi cum se calculează aria unei sfere, căci despre ea este vorba? Se înmulţesc: patru cu trei şi paisprezece şi cu pătratul razei. Cam dificilă această exprimare, nu vi se pare? Să alegem pentru arie şi rază litere, astfel încît să putem rescrie textul ca pe o instrucţiune BASIC-aMIC:

```
20 A=4 * PI * R * R
```

unde A reprezintă aria rezervorului sferic, **PI** este 3,14 iar R=raza rezervorului. 20 este numărul liniei de program.

Observaţie. Atenţie la utilizarea lui " * " pentru înmulţire. Pentru operaţia de ridicare la putere folosiţi " ↑ "

Instrucţiunile:

```
20 A=4 * PI * R * R
```

şi

```
20 A=4 * PI * R ↑ 2
```

sînt echivalente. Vom prefera în continuare ultima formă de scriere.

Ce semnificaţie are linia 20 pentru calculator? În momentul execuţiei acestei linii, calculatorul ia valoarea lui R (trebuie s-o cunoască!), o ridică la pătrat, înmulţeşte rezultatul cu 4 şi apoi cu 3,14 şi, în final, atribuie noul rezultat variabilei A.

Observaţie. Nu putem inversa 4 * PI * R ↑ 2 = A. Această formă de scriere nu are nici un sens pentru calculator.

Pentru identificarea valorilor cunoscute (raza), cît şi pentru cele pe care dorim să le obţinem (aria), putem folosi oricare din cele 26 de litere (A–Z) ale alfabetului. Este bine, însă, să utilizaţi litere cu o anumită semnificaţie, în context cu problema de rezolvat (exemplu: R – rază, A – arie) etc.

În algebra obişnuită, literele alfabetului sînt deseori folosite pentru termeni care pot lua valori numerice variabile. Expresia L+2, de exemplu, poate fi egală cu orice număr L, plus 2.

BASIC-ul are o astfel de structură încît permite utilizatorului să atribuie valori numerice oricărei litere din alfabet.

Dacă aria unui rezervor sferic se calculează cu formula $4\pi R^2$, ne întrebăm cît este aria unui rezervor cu raza de 3 m? Pentru ca aMIC-ul să ne răspundă la această întrebare, introduceţi următoarele linii de program:

```
10 R=3 [RETURN]
20 A=4 * PI * R ↑ 2 [RETURN]
```

Verificaţi cu atenţie programul, după care tastaţi:

RUN [RETURN]

Ce aţi obţinut? Doar **READY?** Într-adevăr, dvs. aveţi dreptate. Dar, de ce numai atît? Nu aţi primit nici un răspuns de la calculator deoarece aţi uitat să-i spuneţi să afişeze valoarea ariei! Ne pare rău! În limbajul BASIC tipărirea valorilor numerice se realizează tot cu instrucţiunea **PRINT.**
Tastaţi:

30 PRINT "A" [RETURN]
RUN [RETURN]

Aţi obţinut 113.097, cît reprezintă aria rezervorului de rază 3 m? Nu!? Dar ce s-a întîmplat? Cum de nu ne-am dat seama de la început că am greşit? Deoarece în linia 30, A apare între ghilimele, la execuţia programului, după cum ştim foarte bine, se va tipări doar A – caracterul cuprins în interiorul ghilimelelor – şi nimic mai mult.

Dacă dorim ca aMIC-ul să tipărească numai litera A, atunci o includem între ghilimele; dacă dorim însă valoarea variabilei A, ca în cazul nostru, nu mai punem ghilimele. Atenţie mărită, deci!
Tastaţi:

30 PRINT A [RETURN]
RUN [RETURN]

şi priviţi cum pe ecranul televizorului a apărut valoarea mult aşteptată: 113.097.

Observaţie. Valorile reale se scriu cu punct zecimal (nu cu virgulă).

Dacă doriţi să afişaţi pe aceeaşi linie atît valoarea lui A cît şi . . . cîteva cuvinte, de exemplu: "A=", tastaţi:

30 PRINT "A=", A [RETURN]

Observaţie. O instrucţiune se poate înlocui, prin tipărirea noii instrucţiuni cu acelaşi număr de linie, cu vechea instrucţiune – lucru pe care l-am dorit şi noi.

```
SCR
10  R=3
20  A= 4*PI*R↑2
RUN
READY
30 PRINT "A"
RUN
A
READY
30  PRINT  A
RUN
113.097
READY
30 PRINT "A=", A
RUN
A=          113.097
READY
■
```

Fig. 2.1. Programul conversaţiei plus rezultate.

Introduceţi din nou comanda **RUN** şi spuneţi-vă părerea în legătură cu rezultatul. Pare mult mai explicit, nu-i aşa?

În fig. 2.1 se prezintă imaginea întregii conversaţii pe care dvs. aţi purtat-o cu calculatorul aMIC.

Variabile numerice

În procesul descrierii programului BASIC pe care l-aţi urmărit au fost introduse noi elemente BASIC, cum ar fi: variabile, instrucţiunea de atribuire şi instrucţiunea **PRINT** de editare a valorilor variabilelor numerice. Variabila numerică poate fi orice literă din alfabet de la A–Z. În general, PC-urile permit folosirea a două litere pentru denumirea variabilelor numerice, de exemplu AB, AC, AE şi aşa mai departe. Alte calculatoare personale folosesc pentru reprezentare, pe lîngă litere şi o cifră cuprinsă între 0 şi 9: A1, A2, ş.a.m.d. Toate aceste posibilităţi pun în evidenţă nu-

Tabelul 2.1

A	AØ	A1	A2	A3	A4	A5	A6	A7	A8	A9
B	BØ	B1	B2	B3	B4	B5	B6	B7	B8	B9
C	CØ	C1	C2	C3	C4	C5	C6	C7	C8	C9
D	DØ	D1	D2	D3	D4	D5	D6	D7	D8	D9
E	EØ	E1	E2	E3	E4	E5	E6	E7	E8	E9
F	FØ	F1	F2	F3	F4	F5	F6	F7	F8	F9
G	GØ	G1	G2	G3	G4	G5	G6	G7	G8	G9
H	HØ	H1	H2	H3	H4	H5	H6	H7	H8	H9
I	IØ	I1	I2	I3	I4	I5	I6	I7	I8	I9
J	JØ	J1	J2	J3	J4	J5	J6	J7	J8	J9
K	KØ	K1	K2	K3	K4	K5	K6	K7	K8	K9
L	LØ	L1	L2	L3	L4	L5	L6	L7	L8	L9
M	MØ	M1	M2	M3	M4	M5	M6	M7	M8	M9
N	NØ	N1	N2	N3	N4	N5	N6	N7	N8	N9
O	OØ	O1	O2	O3	O4	O5	O6	O7	O8	O9
P	PØ	P1	P2	P3	P4	P5	P6	P7	P8	P9
Q	QØ	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
R	RØ	R1	R2	R3	R4	R5	R6	R7	R8	R9
S	SØ	S1	S2	S3	S4	S5	S6	S7	S8	S9
T	TØ	T1	T2	T3	T4	T5	T6	T7	T8	T9
U	UØ	U1	U2	U3	U4	U5	U6	U7	U8	U9
V	VØ	V1	V2	V3	V4	V5	V6	V7	V8	V9
W	WØ	W1	W2	W3	W4	W5	W6	W7	W8	W9
X	XØ	X1	X2	X3	X4	X5	X6	X7	X8	X9
Y	YØ	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
Z	ZØ	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9

mărul mare de alternative pentru reprezentarea variabilelor numerice. Folosirea simplă a literelor alfabetului de la A la Z ne furnizează 26 de variabile, combinațiile de la AA–ZZ oferă încă 26×26=626 variabile, iar folosirea combinațiilor de tipul AØ la Z9 încă 260.

În orice caz, aceasta nu înseamnă că sînteți liberi să folosiți 912 variabile numerice diferite în cadrul aceluiași program. Calculatorul dvs. deține o unitate de memorie ale cărei dimensiuni sînt adesea nesatisfăcătoare pentru înmagazinarea a 912 variabile diferite. Aveți libertatea de a alege dintre aceste 912 variabile, dar numărul de variabile pe care îl puteți folosi o dată este limitat.

Cît privește variabilele numerice utilizate în BASIC-aMIC, ele pot fi **reprezentate fie printr-o literă, fie o literă și o cifră.** Toate variabilele numerice utilizate în BASIC-aMIC sînt reale. Cele 286 de variabile numerice (simple) pe care le puteți utiliza în limbajul BASIC pe calculatorul personal aMIC sînt prezentate în tabelul 2.1.

Instrucțiunea de atribuire LET

Pentru a ilustra conceptul de variabilă cît și funcțiunile instrucțiunii de atribuire, imaginați-vă că, în cadrul programului anterior, există 26 de sertare. Fiecare sertar poate conține în orice moment cîte un număr. Avem deja depozitate numere în cîteva dintre sertare. De exemplu (vezi figura) 8 este în sertarul F,2 este în sertarul B. Ce număr este în sertarul V? Dar în C? 20 este în sertarul . . . . iar 12.5 este în sertarul . . .

A		J	9	T	20
B	2	K	-13.7	U	0
C	6	L	-3.14	V	50
D	7·9	M	100	W	8·25
E	-3	N	10	X	13
F	8	O	60	Y	0
G	80	P	20	Z	-15
H	70	Q	30		
I	12·5	R			
		S	50		

Folosiți creionul pentru a pune 8 în sertarul A. Cu alte cuvinte, scrieți cifra 8 în sertarul liber A. Puneți 14 în R. Puneți 3 în R. Dar, stați! Un sertar poate să conțină la un moment dat un singur număr. Înainte de a-l intro-

duce pe 3, trebuie să-l scoateţi (ştergeţi) din sertar pe 14, pe care l-aţi introdus mai înainte.

A	8
R	3

Cînd calculatorul "pune" un număr într-un sertar, el şterge automat ceea ce a avut anterior în sertar, aşa cum aţi procedat, de altfel, şi dvs. Pentru a pune "3" în sertarul R veţi şterge mai întîi ceea ce a fost anterior în sertar — "14".

Observaţie. Pătrunzînd în interiorul calculatorului personal veţi descoperi una din minunile tehnicii moderne — microprocesorul. Veţi remarca, de asemenea, sertarele din structura "dulapurilor" ce poartă inscripţia "RAM", "EPROM" etc.

Numim A, B, C, . . . , Z variabile. Numerele din A, B, C, . . . , Z reprezintă valorile lui A, B, C, . . . , Z.

Programul nostru foloseşte în linia 10 o instrucţiune de tip **LET** pentru a instrui calculatorul să "pună un număr într-un sertar" sau, altfel spus, să atribuie variabilei R valoarea 3.

În mod analog procedează şi pentru instrucţiunea din linia 20, cu singura deosebire că întîi calculează valoarea expresiei din dreapta semnului egal şi după aceea o atribuie variabilei A.

Sintaxa instrucţiunii. În BASIC-ul Darmouth original, valorile variabilelor numerice sînt stabilite prin introducerea instrucţiunii **LET** a cărei sintaxă este:

Format general
LET <variabilă numerică> = <expresie>

Cuvîntul cheie **LET** semnalează calculatorului că a fost stabilită o relaţie între o variabilă şi un număr. În alte versiuni ale BASIC-ului, inclusiv cele utilizate pentru calculatorul personal aMIC, folosirea lui **LET** este facultativă. Exprimarea **LET** va fi omisă din majoritatea exemplelor BASIC-aMIC din această carte, deşi ea va mai fi folosită uneori pentru a vă reaminti că sistemele mai vechi o folosesc.

În concluzie, ori de cîte ori introducem în memoria calculatorului o linie de forma: variabilă numerică, semnul egal (=) şi apoi o valoare numerică, apăsînd ulterior pe **[RETURN]** spunem de fapt calculatorului să atribuie acea valoare numerică particulară acelei variabile numerice particulare. Altfel spus, în momentul execuţiei instrucţiunii **LET** se evaluează expresia din dreapta semnului "=" după care, valoarea obţinută se atribuie variabilei specificate în partea stîngă a semnului "=".

TEST

În programul anterior, variabila din linia 10 este . . . . . iar valoarea destinată variabilei A din linia 20 este . . . . . .

R

113.097

Particularizînd formatul instrucţiunii de atribuire **LET** pentru linia 20 constataţi că ⟨expresie⟩ este "4 * PI * R ↑ 2" iar ⟨variabilă numerică⟩ este A,

ce reprezintă o zonă de memorie în care se plasează valoarea obţinută (113.097) în urma evaluării expresiei de calcul a ariei rezervorului sferic.

Remarcă. O variabilă definită într-o expresie fără a fi fost definită în prealabil, primeşte automat valoarea zero.

Evaluarea expresiei are loc în conformitate cu următoarele **reguli:**

- se evaluează expresiile cuprinse între paranteze;
- în absenţa parantezelor, ordinea este: operatori unari (exemplu: +A; –A), ridicarea la putere, înmulţire şi împărţire, adunare şi scădere;
- dacă operatorii sînt de aceeaşi prioritate, evaluarea se face întotdeauna de la stînga la dreapta.

Constante numerice. Numerele 4; 3.14 şi 2 (exponentul) reprezintă constante numerice BASIC (v. linia 20). Valoarea acestora este fixă în cadrul unui program – ea nu se modifică în timpul execuţiei programului. Limbajul BASIC recunoaşte trei formate de reprezentare a constantelor numerice: format întreg (exemplu: 4; 2), format real (exemplu: 3.14) şi format exponenţial (vezi Conversaţia 5).

Observaţie. Constanta intrinsecă **PI** poate fi utilizată oriunde sînt permise expresii numerice şi are loc atribuirea valorii 3.14 în locul specificat în expresie.

Aplicaţie. Scrieţi un program BASIC pentru calculul expresiei (1). Se vor prezenta mai multe versiuni.

$$\frac{5/8+2/7}{3/4-1/6} \quad (1)$$

a)
```
10 N=5/8+2/7
20 P=3/4-1/6
30 PRINT N/P
40 END
```

b)
```
10 F=(5/8+2/7)/(3/4-1/6)
20 PRINT F
30 END
```

c)
```
10 PRINT (5/8+2/7)/(3/4-1/6)
20 END
```

Tipărirea valorilor unei variabile numerice

PRINT este, fără îndoială, cea mai utilizată instrucţiune BASIC. Ea permite calculatorului să vă vorbească după cum . . . o programaţi! Formatul general al instrucţiunii este:

Format general
PRINT [<listă>]

unde ⟨listă⟩ reprezintă o succesiune de elemente admise de limbaj, separate între ele prin caracterele "," (virgulă) sau ";" (punct şi virgulă).

Particularizînd formatul instrucţiunii de editare **PRINT** la linia 30

```
30 PRINT A
```

⟨listă⟩ se reduce la variabila A a cărei valoare calculată în linia 20 se afișează în momentul execuției programului. În acest caz, utilizarea separatorilor (" , " sau " ; ") nu a fost necesară.

TEST

a) Completați programul următor atribuind variabilei Y1 valoarea 10 și afișați valoarea variabilei Y1.

```
10 ..........
20 ..........
99 END
- - - - -
20 PRINT Y1
10 Y1=10
99 END
```

b) Precizați funcțiunile următorului program BASIC-aMIC:

```
5 INIT
10 A=5
20 PRINT A
30 B=7
40 PRINT B
50 C=A+B
60 PRINT C
70 PRINT
99 END
- - - - -
```

(5) șterge ecranul; (10) atribuie variabilei A valoarea 5; (20) tipărește valoarea variabilei A; (30) atribuie variabilei B valoarea 7; (40) tipărește valoarea variabilei B; (50) însumează valoarea varibilei A cu valoarea variabilei B și atribuie rezultatul variabilei C; (60) tipărește valoarea variabilei C; (70) tipărește o linie albă; (99) terminare program.

c) Precizați rezultatul următorului program BASIC. Verificați-vă răspunsurile cu un calculator aMIC.

```
c1) 10 LET X=1
    20 LET X=3
    30 PRINT X
    40 END
    RUN
- - - - - - - - -
    3

c2) 10 A=8
    20 B=A
    30 PRINT B
    40 END
    RUN
- - - - - - - - -
    8

c3) 10 A=25.3
    20 PRINT A
    30 A=8
    40 PRINT A
    99 END
    RUN
- - - - - - - - - -
    25.3
    8

c4) 10 A=12
    20 B= 5
    30 PRINT C
    40 END
    RUN
- - - - - - - - - -
    0
```

Separatorul " , " (virgula)

Pentru a merge mai departe cu explicațiile privind utilizarea instrucțiunii **PRINT,** vă propunem acum analiza instrucțiunii introduse ulterior în linia 40:

40 **PRINT** "A=", A

Referindu-ne la sintaxa instrucțiunii **PRINT,** remarcați cum, în acest caz, ⟨listă⟩ este reprezentată de două elemente admise de BASIC și anume: șir de caractere sau **constantă șir** ("A=") și variabilă (A). Cele două elemente sînt separate prin caracterul " , " (virgulă).

Vă reamintiţi cum la execuţia programului, calculatorul a tipărit mai întîi A şi într-o poziţie depărtată (vom vedea îndată cît de depărtată) a afişat 113.097.

Bănuiţi cine a despărţit cele două elemente de pe linia de editare? Aţi intuit desigur că este vorba de virgulă! Atenţie, deci, la virgulă!

Cînd separatorul folosit între elementele unei liste este **virgula**, editarea rezultatelor se face în două zone de 15 caractere fiecare: 1–15 şi 16–30. Dacă valoarea de tipărit depăşeşte lungimea unei zone, elementul următor este tipărit la începutul primei zone libere.

Observaţie. Pentru a afla cîte poziţii are display-ul cu care lucraţi, examinaţi următorul program BASIC şi număraţi apoi X-urile din lungul linii terminal.

```
5 PRINT "         1         2         3         4"
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
99 END
```

În fig. 2.2. puteţi urmări modul în care calculatorul aMIC a executat instrucţiunea **PRINT** din linia 40.

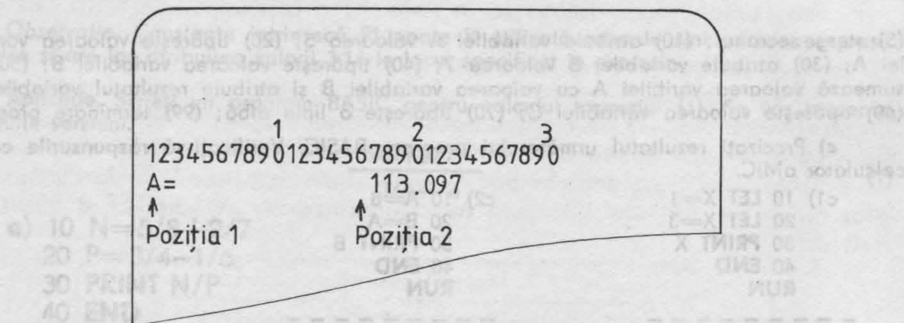
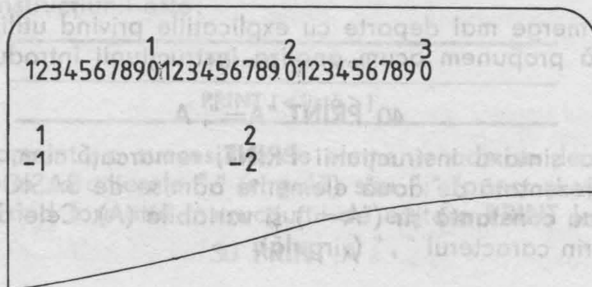


Fig. 2.2. Ilustrarea modului de utilizare a separatorului "," într-o instrucţiune **PRINT** (valori pozitive).

Remarcaţi că valoarea lui A nu s-a afişat din coloana 16 aşa cum am afirmat mai înainte, ci din coloana 17. Explicaţia este următoarea: cînd calculatorul afişează un număr pozitiv sau zero este lăsat mai întîi un spaţiu liber, după care este afişat numărul. Priviţi ce se întîmplă atunci cînd numerele negative sînt afişate alături de cele pozitive.

```
5 INIT            20 PRINT -1, -2
10 PRINT 1, 2     30 END
                  RUN
```



Numerele negative sînt afişate cu semnul minus, fiind urmate de valoarea respectivă, în timp ce numerele pozitive sînt precedate de un spaţiu liber.

TEST

Care este rezultatul execuţiei următoarelor programe BASIC-aMIC?

a)
```
SCR
10 PRINT 1, 2, 3, 4
20 END
RUN
_ _ _ _ _ _ _ _ _ _ _ _
 1              2
 3              4
```

b)
```
SCR
10 PRINT 1, 2, 3, 4, 5, -6, 7
20 END
_ _ _ _ _ _ _ _ _ _ _ _
 1              2
 3              4
 5             -6
 7
```

Separatorul " ; " (punct şi virgulă)

În cazul în care doriţi ca rezultatele pe care le afişează calculatorul dvs. să fie mai apropiate folosiţi în instrucţiunea **PRINT** în loc de virgulă, punct şi virgulă. Utilizarea ca separator a lui " ; " conduce la tipărirea fără salt la următoarea zonă.

Revenind la programul nostru, înlocuiţi în instrucţiunea 40 virgula cu punct şi virgulă.

Tastaţi:

50 PRINT "A="; A [RETURN]

RUN [RETURN]

şi remarcaţi cum imaginea este cu totul alta; 113.097 (valoarea lui A) nu mai este afişat începînd din coloana 16 (17, de fapt, datorită semnului) ci din coloana 4!

Într-o instrucţiune **PRINT** în care folosim ca separator punct şi virgulă numerele pozitive se editează în secvenţa: spaţiu liber, număr, spaţiu liber iar numerele negative după cum urmează: semnul minus, număr, blanc.

Observaţie. Dacă doriţi spaţii libere, le puteţi include în şirurile de caractere respective.

Astfel, la execuţia instrucţiunii:

60 PRINT " A = "; A

după semnul egal, calculatorul va lăsa automat două spaţii libere (blancuri).

Cum răspundeţi la următoarea întrebare? Ce se întîmplă în cazul cînd ⟨listă⟩ din instrucţiunea **PRINT** se termină cu " , " sau " ; "? Iată şi răspunsul. Existenţa unui separator " , " sau " ; " după ultimul caracter al listei suprimă operaţiile RETUR DE CAR şi SALT LA LINIE NOUĂ şi astfel instrucţiunea **PRINT** următoare va edita datele pe aceeaşi linie, în continuare.

Dacă aţi înţeles răspunsul la întrebarea noastră, încercaţi să scrieţi instrucţiunea din linia 60 şi în alt mod. Aţi reuşit? Dacă nu, reflectaţi asupra următoarei versiuni:

60 PRINT " A = ",
65 PRINT A

sau (cu punct şi virgulă):

```
60 PRINT " A = ";
65 PRINT A
```

Restrîngerea unui program

Îndemînarea în programarea calculatoarelor nu constă atît în cunoaşterea unui număr foarte mare de instrucţiuni (comenzi) diferite, cît în găsirea unor căi de a scrie programe eficiente. Programele eficiente sînt acele programe care îşi ating scopul propus prin folosirea unui număr minim de linii de program.

Priviţi la lista programului. Acest program foloseşte 6 linii de instrucţiuni. Deşi numărul acestora nu este impresionant de mare, vă punem următoarea întrebare: este posibilă restrîngerea acestui program? Încercaţi această secvenţă de instrucţiuni BASIC:

```
20        [RETURN]
30 PRINT "A = "; 4 * PI * R ↑ 2   [RETURN]
```

Considerînd că acest program de şase linii se află în memoria calculatorului dvs. modificaţi programul prin tipărirea numărului liniei 20 şi apăsarea tastei **[RETURN]**. Dacă vă amintiţi, din conversaţia precedentă, acest tip de operaţie şterge din memoria calculatorului linia respectivă. După aceea, veţi rescrie linia 30, înlocuind variabila numerică A cu operaţia 4 * PI * R ↑ 2. Variabila numerică A este acum complet eliminată din prima parte a programului (linia 20).

Tastaţi:

RUN [RETURN]

Se pare că aMIC-ul a afişat aceeaşi valoare ca şi în versiunea precedentă.

Observaţie. Instrucţiunea **PRINT** admite ca element al listei şi expresie numerică.

Dar, această versiune de program foloseşte doar cinci linii de program şi conţine o variabilă mai puţin. Aceasta nu reprezintă o reducere remarcabilă a complexităţii programului, dar exemplul ilustrează că este într-adevăr posibil să rezolvi o problemă în cel puţin două moduri diferite.

Majoritatea problemelor pot fi rezolvate în nenumărate moduri. Dar întotdeauna există doar cîteva moduri mai eficiente pentru rezolvarea acestora; şi aici intervin priceperea şi experienţa dvs.

TEST

Precizaţi care sînt rezultatele execuţiei următoarelor programe BASIC-aMIC?

a)	b)	c)
10 A=2	10 A=2	10 C=A+B
20 B=3	20 B=3	20 A=2
30 **PRINT** A+B	30 C=A+B	30 B=3
40 **END**	40 **PRINT** C	40 **PRINT** C
RUN	50 **END**	50 **END**
	RUN	**RUN**
....	....	....
5	5	0

SUMAR

Instrucţiunea **PRINT**

Instrucţiune	Exemplu
PRINT ⟨şir de caractere⟩	10 **PRINT** "A+B"
	5 A=7
PRINT ⟨variabilă numerică⟩	10 **PRINT** A
PRINT ⟨expresie numerică⟩	10 **PRINT** 2 * 3
PRINT ⟨listă⟩	10 **PRINT** A; "B="; 4*3.2.

In procesul exemplificării acestor instrucţiuni **PRINT**, a fost necesar să introducem operaţiile de atribuire de tip **LET.**

TEST

1. Care dintre următoarele instrucţiuni **PRINT** trebuie precedate de o operaţie de atribuire?

a) **PRINT** "DIMINEŢI CU FERESTRE DESCHISE"
b) **PRINT** "5+3"
c) **PRINT** "3·14TZURCA"
d) **PRINT** 5 * 2–3
e) **PRINT** Y3
f) **PRINT** A+Z4
g) **PRINT** "A+Z4"
h) **PRINT** "A$"

2. Care este răspunsul calculatorului pentru fiecare din următoarele programe?

a)
```
10 INIT
20 PRINT 5 * 8
30 END
```
b)
```
10 INIT
20 PRINT 5 * M
30 PRINT
```
c)
```
10 INIT
20 X=2
30 Y=3
40 PRINT "R="; X * Y
50 END
```
d)
```
10 INIT
20 A=2
30 B=5
40 PRINT A; " * "; B
50 PRINT "R="; A * B
60 END
```

□ Particularităţi ale programării în limbajul BASIC-PRAE

vol. 2, pag. 199

Priviţi, vă rugăm, programul. Examinaţi-l linie cu linie

```
10 R=3
20 A=4 * PI * R ↑ 2
30 PRINT A
40 PRINT "A=", A
50 PRINT "A="; A
60 PRINT 4; " * "; PI; " * "; R;
   " ↑"; 2; "="; A
70 PRINT 4; " * "; PI; " * "; R;
   "↑"; 2; "="; 4 * PI * R ↑ 2
80 END
```

după modelul:

10 Atribuie variabilei R valoarea 3.
20 Atribuie variabilei A valoarea expresiei numerice $4\pi R^2$.
30 Tipăreşte valoarea calculată în linia precedentă.
40 Virgula decalează imprimarea valorii variabilei A la începutul zonei următoare. Linia terminal se consideră împărţită în zone a 14

caractere (5 cîmpuri pentru un terminal cu 80 caractere, respectiv 8 cîmpuri pentru linia cu 132 caractere).

50 Punctul şi virgula determină imprimarea valorii lui A imediat după semnul egal.

60, 70 Editează valoarea ariei rezervorului (A) calculată în două moduri diferite.

Remarci

- Numele variabilelor simple sînt formate dintr-o literă urmată de oricîte caractere alfanumerice (numai primele două caractere sînt luate în considerare).
- Numele variabilelor încep obligatoriu cu o literă şi nu trebuie să coincidă cu cuvintele rezervate.
- Variabilele pot fi numerice şi variabile şir.
- În momentul lansării în execuţie a unui program, toate variabilele acestuia sînt iniţializate cu zero.
- Într-un program BASIC-PRAE este posibilă definirea prin acelaşi nume a unei variabile simple şi a unei variabile indexate.
- Sînt permise două tipuri de constante: numerice şi şiruri.
- Toate constantele BASIC-PRAE sînt considerate în format flotant.
- Constantele se pot reprezenta într-unul din formatele: întreg, real, exponenţial.
- Operaţiile aritmetice admise sînt: adunarea (+), scăderea (–), înmulţirea (*), împărţirea (/), ridicarea la putere (↑).
- Dacă într-o expresie aritmetică se folosesc variabile cărora nu li s-a atribuit anterior o valoare, expresia rămîne nedefinită.
- Prioritatea operaţiilor este cea cunoscută.
- BASIC-PRAE acceptă mai multe instrucţiuni separate între ele prin caracterul " : " pe o linie program linie multiinstrucţiune (exemplu: 30 R=3 : A=4 * PI * R2 : **PRINT** A).
- Instrucţiunea de atribuire **LET** (cuvîntul **LET** este opţional) poate fi plasată oriunde în cadrul unei linii multiinstrucţiune. (Exemplu: 10 **LET** A=3 : B=5 : VIN=18).



În continuare, tastaţi **NEW, [CR]** în vederea introducerii programului analizat în memoria calculatorului personal PRAE. După ce v-aţi asigurat că toate liniile au fost introduse corect, executaţi programul cu comanda **RUN, [CR]**. Analizaţi rezultatele.

Aplicaţie. Fiind date coordonatele punctelor M (3, 4) şi N (–3.4, 5.75), să se scrie un program care calculează şi afişează valoarea distanţei MN.

```
 5 CLS
10 X1=3 : Y1=4
20 X2=–3 . 4 : Y2=5. 75
30 MN=((X2–X1) ↑ 2+(Y2–Y1) ↑ 2) ↑ 0 . 5
40 PRINT "DISTANTA MN=", MN
50 END
```

TESTE

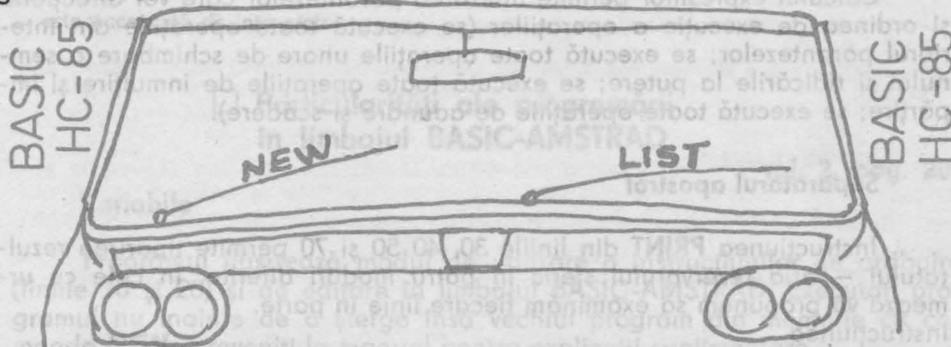
- Se consideră dreptunghiul cu vîrfurile în punctele M(1,5); N $\left(\frac{2}{5}, \frac{5}{19}\right)$; P(2,3); Q$\left(\frac{13}{5}, \frac{21}{5}\right)$.
  Să se scrie un program BASIC care calculează şi afişează lungimile laturilor şi ale diagonalelor.
- Să se scrie un program BASIC care calculează şi afişează valorile expresiilor:
  a) $E=\frac{2,83\,(3,37)}{7,82}+3,29$
  b) $X=\sqrt{183,5}$
  c) $A=\sqrt{p(p-2)\,(p-3)\,(p-7)}$
  unde, p=6.
- Să se calculeze latura cubului (l) înscris într-un con circular drept cu raza bazei R=3 m şi înălţimea h=15 m. Se va utiliza relaţia: $l=\frac{2Rh}{2R+h\sqrt{2}}$.
- Să se calculeze muchia unui tetraedru regulat înscris în sfera de rază R=3 m. Se va utiliza relaţia: $X=\frac{2R\sqrt{6}}{3}$.
- Să se calculeze latura (l) unui cub înscris în sfera de rază R=3 m. Se va utiliza relaţia: $l=\frac{2R\sqrt{3}}{3}$.

☐ Particularităţi ale programării în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 199

Variabile

Înainte de a studia utilizarea instrucţiunilor **LET** şi **PRINT** în limbajul BASIC implementat pe calculatoarele HC-85, TIM S, SPECTRUM, ştergeţi cu comanda **NEW** orice program sau variabilă anterioară şi porniţi totul de la zero. Aşadar, tastaţi **NEW** (tasta A), **[ENTER]**, **[LIST]**, **[ENTER]** şi vă veţi convinge că în memoria calculatorului dvs. nu mai există nici un program BASIC.



Introduceţi acum programul şi apoi executaţi-l. Nu uitaţi că după introducerea fiecărei linii trebuie să tastaţi **[ENTER]**.

```
10 LET R=3
20 LET A=4 * PI* R ↑ 2
30 PRINT A
40 PRINT "A=", A
50 PRINT "A="; A
60 PRINT "A" ' A
70 PRINT "4 * 3 . 14 *"; R; " ↑ ";
   2; "="; A
80 PRINT 4; " * "; PI; " * "; R;
   " ↑ "; 2; "="; 4 * PI * R ↑ 2
9999 STOP
```

Instrucțiunile care prezintă interes pentru noi sînt cele din liniile 10, 20, 40, 50, 60. Restul instrucțiunilor (liniile 30, 70 și 80) sînt cunoștințe mai vechi.

Instrucțiunea **LET** din linia 10 atribuie variabilei R valoarea 3. Cuvîntul **LET** este obligatoriu (tasta L), deci nu poate fi omis din structura nici unei instrucțiuni de atribuire.

Cît privește variabilele BASIC HC-85, TIM S, SPECTRUM **regulile** sînt următoarele:

- numele variabilelor numerice simple pot fi de lungime arbitrară pornind de la o literă (obligatoriu, primul caracter) și continuînd cu litere, cifre, spațiu;
- numele variabilelor se referă la înțelesul literelor și nu la modul lor de scriere (litere mari, litere mici sau combinat).

Instrucțiunea LET

În linia 20, instrucțiunea **LET** atribuie valoarea expresiei ce calculează aria rezervorului sferic variabilei A. Cu această ocazie spunem că am definit variabila A (în linia 10 am definit variabila R).

Rețin atenția operatorii aritmetici pe care i-am utilizat în expresia numerică: " * " (pentru înmulțire) și " ↑ " (pentru ridicare la putere).

La întîlnirea constantei intrinseci **PI** se va atribui valoarea 3.14 . . . în locul specificat în expresie.

În BASIC HC-85, TIM-S, SPECTRUM sînt folosiți următorii operatori aritmetici: " ↑ " (ridicare la putere), " * " (înmulțire), " / " (împărțire), " + " (adunare), " − " (scădere), " − " (schimbarea semnului).

Observație. Pentru ridicarea la putere este necesar ca puterea să fie număr pozitiv.

Calculul expresiilor permite utilizarea parantezelor care vor direcționa și ordinea de execuție a operațiilor (se execută toate operațiile din interiorul parantezelor; se execută toate operațiile unare de schimbare a semnului și ridicările la putere; se execută toate operațiile de înmulțire și împărțire; se execută toate operațiile de adunare și scădere).

Separatorul apostrof

Instrucțiunea **PRINT** din liniile 30, 40 50 și 70 permite tipărirea rezultatului – aria rezervorului sferic în patru moduri diferite. În cele ce urmează vă propunem să examinăm fiecare linie în parte.
Instrucțiunea

```
30 PRINT A
```

tipărește valoarea variabilei A calculată anterior 113.09734.

Instrucţiunea **PRINT** din linia 40 foloseşte separatorul " , " (virgulă) care, în BASIC HC-85, TIM S, SPECTRUM determină ca tipărirea valorilor să se facă cîte două pe linie, una la începutul liniei iar cealaltă la mijloc.

Separatorul " ; " (punct şi virgulă) utilizat în cadrul instrucţiunii **PRINT** din linia 50 nu are nici un efect special, valorile fiind tipărite una după alta, pe aceeaşi linie.

Şi acum, un element de noutate pentru BASIC HC-85, TIM S, SPECTRUM – utilizarea apostrofului (v. linia 60) ca separator al elementelor unei liste din cadrul instrucţiunii **PRINT.**

Caracterul apostrof face ca valorile să fie tipărite una sub alta.

TEST

Examinaţi următorul program, linie cu linie şi comentaţi fiecare instrucţiune:

```
10 PRINT                70 PRINT 2, 3
20 PRINT                80 PRINT, 2
30 PRINT                90 PRINT, 2; 3
40 PRINT "DO"; 40      100 PRINT 2 ' A
50 PRINT, "DO"; 50      15 PRINT "12345678901234567890
60 PRINT 23            123"
```

Linie multiinstrucţiune. Programul poate fi scris şi sub forma liniilor multiinstrucţiune. Concatenarea instrucţiunilor se face cu ajutorul simbolului " : ".

TEST

Răspundeţi prin DA sau NU la următoarele întrebări:

– Numele variabilelor numerice simple sînt formate dintr-un număr limitat de caractere.
– Numele variabilelor încep obligatoriu cu o literă.
– Litera mare şi litera mică omoloagă sînt interpretate la fel.
– DANI şi dani desemnează aceeaşi variabilă.
– Variabila aG 1 este scrisă corect.
– Atribuirea de noi valori variabilei se face prin instrucţiunea **LET.**
– Cuvîntul **LET** în cadrul unei instrucţiuni de atribuire în BASIC HC-85, TIM S, SPECTRUM este opţional.
– Operatorul aritmetic pentru ridicarea la putere este ✻✻.
– Expresia E=Y ↑ – 3 este corectă din punct de vedere BASIC.
– La scrierea expresiei
  4 " ✻ " PI " ✻ " R " ↑ " 2 " = " 4 ✻ **PI** ✻ R ↑ 2
  interpretorul BASIC HC-85, TIM S, SPECTRUM nu semnalează eroare.
– Instrucţiunea de atribuire
  10 **LET** = 5
  este acceptată de interpretor.

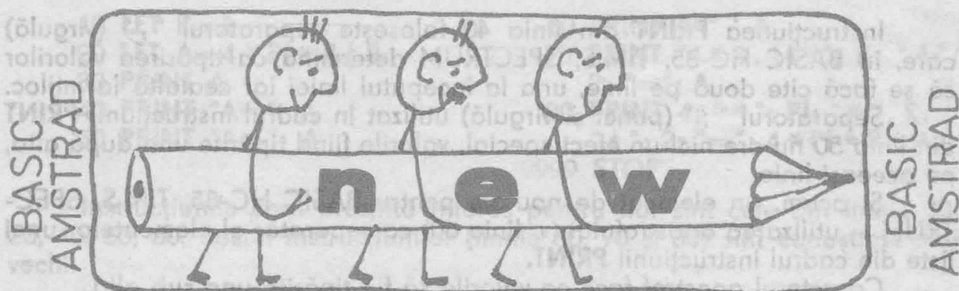
☐ Particularităţi ale programării în limbajul BASIC-AMSTRAD

vol. 2, pag. 200

Variabile

Programul ilustrează modul de utilizare a instrucţiunilor de atribuire (liniile 10 şi 20) şi de editare în limbajul BASIC-AMSTRAD. Executaţi programul nu înainte de a şterge însă vechiul program din memorie cu comanda **NEW** şi reveniţi la manual pentru explicaţii suplimentare.

Linia 10 determină atribuirea unei valori – 3 în cazul nostru – variabilei reale r. O variabilă este identificată printr-un şir de carac-



tere care începe cu o literă, urmată de un număr oarecare de litere sau cifre. Variabilele admise de limbajul BASIC-AMSTRAD sînt de următoarele tipuri: numerice reale, numerice întregi, șiruri de caractere.

Remarci

- Numele de variabilă nu trebuie să coincidă cu cuvintele rezervate BASIC-AMSTRAD.
- Fiecare nume de variabilă dintr-un program este asociat unei zone din memorie ce este utilizată pentru stocarea valorii curente a variabilei.
- Tipul unei variabile se poate stabili și prin caracterele speciale:
  % – variabile numerice întregi
  ! – variabile numerice reale
  $ – variabile șir de caractere
  ce se atașează numelui variabilei.
- Orice variabilă numerică la care ultimul caracter din nume nu este simbolul "%" este considerată variabilă pentru date în virgulă mobilă, adică variabila reală (variabilele R, A din program).
- Numele unei variabile numerice reale și numele unei variabile numerice întregi desemnează întotdeauna variabile distincte, chiar dacă numele este același, diferind, evident, numai prin simbolul "%" ce trebuie să fie prezent în cazul variabilei întregi (v. conversația 3).
- O variabilă poate fi declarată și implicit cu ajutorul instrucțiunilor: **DEFINT, DEFREAL, DEFSTR** (v. vol. 2, pag. 12).

TEST

Răspundeți prin DA sau NU la următoarele întrebări:
- Variabila R este de tip întreg.
- Variabila R% este de tip real.
- Variabilele A1% și A1 au același nume dar sînt de tipuri diferite.
- Într-un program pot exista variabile simple numerice și variabile simple nenumerice (șir de caractere) cu același nume: L, L%, L$.
- Următoarele nume de variabile numerice sînt corecte: AGA . 4%, AGA – 4%, A – G – A – 4%.
- Următoarele nume de variabile numerice sînt incorecte: 38X%, 3 – 8X%, A38 A, LET, TEL, ELT.

Operații numerice

În BASIC-AMSTRAD datele numerice sînt prelucrate cu ajutorul operatorilor și funcțiilor (v. vol. 2, pag. 12) numerice.
În linia 20 a programului

```
20 a=4 * PI * r ↑ 2
```

s-au utilizat: operatorul de atribuire "=" ce asignează valoarea expresiei numerice ($4\pi R^2$) variabilei a, operatorii aritmetici " * " și " ↑ " pentru înmulțire, respectiv ridicare la putere și constanta intrinsecă **PI**.

Remarci

- O instrucţiune de atribuire memorează valoarea unei expresii plasate în partea dreaptă a semnului de egalitate ("=") în zonele de memorie corespunzînd variabilei specificate în partea stîngă a semnului de egalitate ("=").
- Structura generală a unei instrucţiuni de atribuire în BASIC-AMSTRAD cuprinde cuvîntul opţional **LET**, urmat în mod obligatoriu de numele unei variabile, de semnul de egalitate ("=") şi de o expresie numerică.
- În BASIC-AMSTRAD sînt folosiţi următorii operatori aritmetici: " ↑ " (ridicarea la putere), " * " (înmulţire), " / " (împărţire), " + " (adunare), " – " (scădere), " – " (schimbarea semnului).
- Constanta intrinsecă **PI** poate fi utilizată oriunde sînt permise expresii numerice şi are loc atribuirea valorii 3.14519265 în locul specificat în expresie.
- Ordinea de execuţie a operaţiilor într-o expresie numerică este: 1) se execută toate operaţiile din interiorul parantezelor; 2) se execută toate operaţiile unare de schimbare a semnului (–) şi ridicările la putere (↑) – de la stînga spre dreapta; 3) se execută toate operaţiile de înmulţire şi împărţire (de la stînga spre dreapta); 4) se execută toate operaţiile de adunare şi scădere (de la stînga spre dreapta).
- Orice operator aritmetic trebuie urmat de o expresie numerică.
- În cazul operaţiilor de ridicare la putere a unor numere negative, nu este obligatorie folosirea parantezelor.
- Atunci cînd într-o expresie numerică nu există paranteze dar operatorii sînt de aceeaşi prioritate în ierarhia execuţiei, evaluarea se face întotdeauna de la stînga la dreapta.
- Ordinea normală în care este evaluată o expresie poate fi schimbată prin folosirea parantezelor.

TEST

Răspundeţi prin DA sau NU la următoarele întrebări:

– Expresia X = B ↑ – 2 este scrisă corect.
– Expresia X = – B ↑ 2 este scrisă corect.
– Expresia X=–B ↑ A este scrisă corect.
– Expresia X = (– B) ↑ 2 este scrisă corect.
– Expresia X = 3 + – 4 este scrisă corect.
– Expresia X = A (– B) este scrisă corect.
– Expresia X = – B + A * – C este scrisă corect.
– În expresia X = X * Y / Z se înmulţesc mai întîi X cu Y iar produsul se împarte după aceea la Z.
– În expresia E = X * (Y / Z) se înmulţeşte mai întîi X cu Y iar produsul se împarte după aceea la Z.
– În expresia E=(X+Y) * V se însumează mai întîi Y cu X şi apoi rezultatul se înmulţeşte cu V.
– În cadrul expresiei E = X + (Y – ((A / B) ↑ 2)) se împarte mai întîi A la B, apoi se face ridicarea la putere, urmată de diferenţă; în final, se adună X la valoarea dinainte.
– În cadrul expresiei:

E = 4 * PI * R ↑ 2

se efectuează mai întîi ridicarea la putere (R ↑ 2), după aceea se înmulţeşte rezultatul cu PI, iar în final se înmulţeşte noul rezultat cu 4.
– În cadrul expresiei:

E = 4 * PI * R ↑ 3/ 3

se efectuează mai întîi ridicarea la putere (R ↑ 3), după aceea se împarte rezultatul la 3, se înmulţeşte în continuare rezultatul obţinut cu 4 iar în final noul rezultat se amplifică cu PI.

Tabulare

După cum aţi putut constata, o expresie numerică poate să conţină o singură variabilă sau constantă sau un grup de variabile sau constante şi funcţii, toate numerice, reunite prin operatori numerici. După ce expresiile sînt evaluate, valorile lor se atribuie, prin instrucţiunea de atribuire

LET, unor variabile. După evaluare, ele sînt tipărite (la imprimantă) sau afişate (la terminal) cu ajutorul instrucţiunii **PRINT** (vezi liniile 25, 30, 35, 40, 45).

```
25 PRINT A
30 PRINT "A=", A
35 PRINT "A=; A
40 PRINT 4; "X"; PI; "X"; R; "↑"; "2"; "="; A
45 PRINT 4; "X"; PI; "X"; R; "↑"; 2; "="; 4 * PI * R ↑ 2
```

Formatul general al instrucţiunii **PRINT** este:

FORMAT GENERAL

```
<număr de linie> PRINT [# <număr canal>,] [<listă>]
```

în care:

⟨număr canal⟩ este numărul canalului necesar desemnării unei ferestre ecran, imprimantei, casetei sau dischetei pe care textul urmează să fie imprimat. Valoarea implicită pentru ⟨număr canal⟩ se ia zero (monitorul);

⟨listă⟩ reprezintă o succesiune de elemente admise de limbaj (constante, variabile, expresii numerice etc.) separate între ele prin caracterele " , " sau " ; ".

```
113.097336
A=            113.097336
A= 113.097336
4× 3.14159265× 3^2=113.097336
4× 3.14159265×3^2=113.097336
```

Fig. 2.3. Rezultatele execuţiei programului.

Ce se întîmplă în urma execuţiei instrucţiunii **PRINT** din linia 25? După cum remarcaţi (v. fig. 2.3), pe monitor s-a afişat simplu 113.097336 ce reprezintă (cum numai noi ştim) valoarea ariei rezervorului sferic.

Parcurgînd în continuare rezultatele obţinute, remarcaţi pe rîndul imediat următor (v. fig. 2.3) aceeaşi valoare dar precedată şi de numele variabilei. Acest rezultat se datorează instrucţiunii **PRINT** din linia 30 care utilizează separatorul " , " (v. formatul general).

În BASIC-AMSTRAD, virgula cere calculatorului să afişeze valoarea următoare peste 13 coloane. Totuşi, dacă numărul de caractere ce trebuie afişat depăşeşte cifra 12, valoarea următoare este decalată cu alte 13 coloane cu scopul de a păstra totdeauna un spaţiu între valorile de afişat.

Observaţie. Mărimea zonei de 13 caractere poate fi modificată prin instrucţiunea **ZONE** (vezi Conversaţia 3).

Pe rîndul imediat următor (vezi fig. 2.3) remarcaţi, evident, acelaşi rezultat dar altfel editat. De această dată, în cadrul instrucţiunii **PRINT** (vezi

linia 35) am utilizat simbolul " ; " care comandă calculatorului să afişeze următoarea valoare imediat după precedenta (numai să nu fie prea mare pentru a se menţine pe linie).

Următoarele valori (vezi fig. 2.3) reprezintă tot valoarea ariei rezervorului sferic, calculată însă în două moduri (vezi liniile 40 şi 45) diferite.

Linie multiinstrucţiune (multiplă)

În BASIC-AMSTRAD orice linie de program poate conţine un număr arbitrar de instrucţiuni (linie multiinstrucţiune sau multiplă) separate între ele cu ajutorul caracterului " : " (două puncte).

Încercaţi, în continuare, să modificaţi programul utilizînd liniile multiinstrucţiune. Ştergeţi programul precedent utilizînd comanda **NEW** şi introduceţi următoarea secvenţă de instrucţiuni.

```
5 MODE 2
10 r=3 : a=4 * PI * r ↑ 2
20 PRINT a : PRINT 4; "x"; PI; "x"; r; " ↑ "; 2;
   "="; a : PRINT 4; "x"; PI; "x"; r; " ↑ "; 2;
   "="; 4 * PI * r ↑ 2
```

Executaţi programul şi analizaţi rezultatele obţinute. Ele trebuie să fie identice cu cele ale programului precedent.

□ Particularităţi ale programării în limbajul BASIC-COMMODORE

vol. 2, pag. 200

Să analizăm împreună programul linie cu linie.

```
10 R=3                    50 ? "A="; A
20 A=4 * π * R ↑ 2        60 ? 4 "x" π "x" R " ↑ " 2 "=" A
30 ? A                    70 ? 4 "x" π "x" R " ↑ " 2 "="
40 ? "A=", A                 4 * π * R ↑ 2
```

10 Atribuie variabilei R valoarea 3.
20 Atribuie variabilei A valoarea expresiei numerice $4\pi R^2$.
30 Tipăreşte valoarea calculată în linia 20.
40 Virgula decalează imprimarea valorii variabilei A la începutul zonei următoare. Linia terminal se consideră împărţită în 4 zone a cîte 10 caractere fiecare.
50 Punctul şi virgula determină imprimarea valorii lui A imediat după semnul egal.
60, 70 Editează valoarea ariei rezervorului calculată în două moduri diferite.

Remarci

- Numele variabilelor simple sînt formate dintr-o literă urmată de oricîte caractere alfanumerice (numai primele două caractere sînt luate în considerare).
- Numele unei variabile nu trebuie să coincidă cu cuvintele rezervate.
- Variabilele pot fi: numerice reale, numerice întregi, şir.

- Tipul întreg al unei variabile se indică prin caracterul "%" adăugat numelui acesteia.
- Într-un program pot exista variabile simple numerice (tip întreg, real) şi variabile simple nenumerice cu acelaşi nume.
- BASIC-COMMODORE acceptă constante numerice (întregi, reale) şi constante şir.
- Constantele numerice pot fi reprezentate în format: întreg, zecimal, exponenţial.
- Operaţiile aritmetice admise sînt: adunarea (+), scăderea (−), înmulţirea (*), împărţirea (/), ridicarea la putere (↑).
- Prioritatea operaţiilor este cea cunoscută.
- Dacă într-o expresie aritmetică se folosesc variabile cărora nu li s-a atribuit anterior o valoare, expresia rămîne nedefinită.
- BASIC-COMMODORE acceptă liniile multiinstrucţiune, instrucţiunile fiind separate de caracterul " : ".
- Într-o instrucţiune de atribuire cuvîntul **LET** este opţional.

COMMODORE new RUN COMMODORE

Şi-acum, eliminaţi programul curent din memoria calculatorului cu comanda **NEW, [RETURN]**, tastaţi programul şi dacă nu aveţi erori de sintaxă, executaţi programul cu comanda **RUN, [RETURN]**. Aţi obţinut pe monitorul dvs. 113.097336? Aţi înţeles foarte bine semnificaţia celor doi separatori " , " şi " ; " din cadrul unei instrucţiuni **PRINT**? Dacă da, vă propunem să realizaţi în continuare, singuri, fără ajutorul nostru, o simulare privind execuţia acestui program. Vă sugerăm să folosiţi o foaie de matematică pe care s-o împărţiţi în 40 de coloane (0÷39). Atenţie la cele patru zone ale liniei terminal! Cel care vă poate verifica cel mai bine este numai calculatorul dvs. personal COMMODORE. Succes!

Aplicaţie. Fiind date punctele A(3,4) şi B(5,6) să se scrie un program care determină ecuaţia dreptei definită de cele două puncte.

```
5 CHR$ (147)
10 X1 = 3 : Y1 = 4
20 X2 = 5 : Y2 = 6
30 PANTA = (Y2 − Y1) / (X2 − X1)
40 TLIBER = Y1 − M * X1
50 PRINT " ( "; X1; " , "; Y1; "), ("; X2;
   " , "; Y2; " ) "
60 PRINT "ECUAŢIA DREPTEI ESTE: Y = ";
70 PRINT PANTA; " * X + ( "; TLIBER;
   " ) "
80 END
```

TESTE

- Să se calculeze viteza şi timpul de cădere ale unui corp care cade liber (fără viteză iniţială) de la înălţimea de 30 m. Se vor utiliza relaţiile de calcul: $v=\sqrt{2gh}$ şi $t=\sqrt{\frac{2h}{g}}$.
- Să se calculeze timpul de urcare şi înălţimea maximă la care se ridică un corp care este aruncat vertical cu viteza $v_0=20$ m/s. Se vor utiliza relaţiile de calcul: $t=\frac{v_0}{g}$ şi $h=\frac{v_0^2}{2g}$, unde $g=9{,}81$ m/s².

- Două corpuri de mase $m_1=1$ kg şi $m_2=2$ kg se mişcă unul spre celălalt cu vitezele $v_1=2$ m/s şi $v_2=-3$ m/s. Să se calculeze energia cinetică pierdută prin ciocnirea lor (plastică). Se va utiliza relaţia: $Q=\frac{1}{2}\frac{m_1\cdot m_2}{m_1+m_2}(v_1-v_2)^2$ J.
- Să se calculeze presiunea datorată apei de mare la o adîncime h=80 cm. Se va utiliza relaţia: $p=\rho gh$, unde $\rho=1{,}026$ g/cm^3.
- Să se calculeze viteza de scurgere a apei printr-un tub cu diametrul interior d=4 cm, ştiind că debitul de volum Qv=10 litri/minut. Se va utiliza relaţia: $v=40v/\pi d^2$ (m/s).

□ Particularităţi ale programării în limbajul BASIC-80

vol. 2, pag. 200

Analiza programului

Priviţi programul. Vă invităm să-l lecturaţi cu atenţie încercînd, pe cît posibil, o comparaţie cu programele precedente.

```
10 PI=3.14159
20 R=3
30 A=4 * PI * R ↑ 2
40 PRINT A
50 PRINT "A=", A
60 PRINT "A="; A
70 PRINT 4 " * " PI " * " R " ↑ " 2
   "=" A
80 PRINT 4 " * " PI " * " R " ↑ " 2
   "=" 4 * PI * R ↑ 2
```

Aţi găsit vreo deosebire? Vom examina acest program linie cu linie.

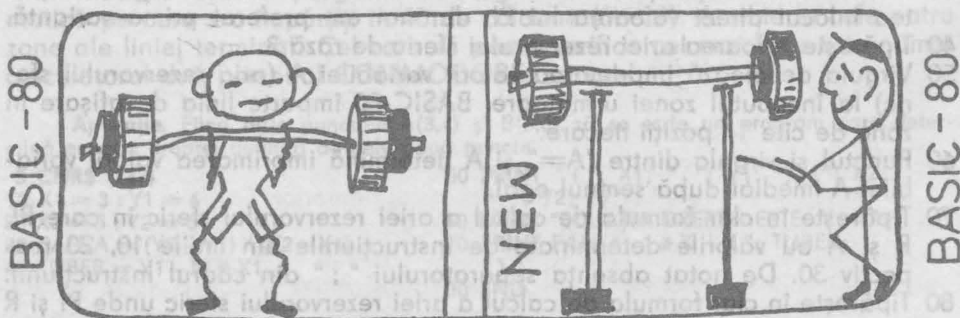
10 Atribuie valoarea 3.14159 variabilei PI ! În BASIC-80 lipseşte constanta π şi, în consecinţă, o vom transmite noi microcalculatorului. Numele variabilei conţine două litere.
20 Iniţializează pe R cu valoarea 3.
30 Atribuie valoarea expresiei numerice $4\pi R^2$ variabilei A. Desigur, se putea înlocui direct valoarea lui PI, dar noi am preferat prima variantă.
40 Tipăreşte valoarea ariei rezervorului sferic de rază 3.
50 Virgula decalează imprimarea valorii variabilei A (aria rezervorului sferic) la începutul zonei următoare. BASIC-80 împarte linia de afişare în zone de cîte 14 poziţii fiecare.
60 Punctul şi virgula dintre "A=" şi A determină imprimarea valorii variabilei A imediat după semnul egal.
70 Tipăreşte în clar formula de calcul a ariei rezervorului sferic în care PI, R şi A au valorile determinate de instrucţiunile din liniile 10, 20 respectiv 30. De notat absenţa separatorului " ; " din cadrul instrucţiunii.
80 Tipăreşte în clar formula de calcul a ariei rezervorului sferic unde PI şi R au valorile determinate de instrucţiunile din liniile 10 şi 20 iar A se calculează chiar în cadrul instrucţiunii. Nu este necesară utilizarea separatorului " ; ".

Remarci

- În BASIC-80 nu sînt restricţii privind lungimea numelui unei variabile dar numai primele 40 de caractere sînt luate în considerare.
- Numele unei variabile nu trebuie să coincidă cu un nume rezervat (comenzi, instrucţiuni, funcţii, operatori logici). El poate conţine litere, cifre sau caracterul " . ".

- Variabilele pot fi variabile întregi, simplă precizie sau dublă precizie și variabile șir.
- Tipul unei variabile poate fi stabilit și printr-un caracter special ce urmează numele variabilei: $ – variabila de tip șir; % – variabilă întreagă; ! – variabilă simplă precizie; # – variabilă dublă precizie.
- Tipul unei variabile BASIC-80 poate fi declarat și explicit prin instrucțiunile **DEFINT, DEFSNG, DEFDBL.**
- Tipul implicit pentru o variabilă este simplă precizie.
- Sînt permise două tipuri de constante: numerice și șiruri.
- BASIC-80 recunoaște cinci tipuri de constante numerice: întregi, virgulă fixă, virgulă mobilă (forma cu exponent), virgulă mobilă dublă precizie, hexazecimale și octale.
- Constantele pot fi reprezentate în simplă precizie (memorate cu 7 cifre și reprezentate cu max. 6 cifre) și dublă precizie (cu 16 cifre).
- Operațiile aritmetice admise sînt: adunarea (+), scăderea (–), înmulțirea (*), împărțirea (/), ridicarea la putere (↑), împărțirea întreagă (\) și restul împărțirii întregi (**MOD**).
- Dacă se atribuie unei variabile numerice o constantă al cărei tip diferă de cel al variabilei, atunci valoarea numerică se convertește automat la tipul variabilei.
- La evaluarea expresiilor, toți operanzii sînt convertiți automat la tipul operandului de precizie maximă care este și tipul rezultatului expresiei.
- La conversia din virgulă mobilă în întreg, se face rotunjirea la întregul cel mai apropiat și nu se trunchiază partea fracționară.
- La conversia din dublă precizie în simplă precizie se rețin numai primele 7 cifre zecimale, cu rotunjire la ultima cifră.
- Operatorii aritmetici (\) și **MOD** se pot folosi și cu operanzi neîntregi ce sînt automat convertiți la întregi înainte de operație.
- Prioritatea operațiilor este cea cunoscută.
- În cazul unei depășiri sau împărțiri cu zero se afișează un mesaj, iar execuția programului continuă.

Dacă ați înțeles programul puteți, desigur, să-l tastați și apoi să-l executați. Înainte de a-l tasta, dați însă comanda **NEW** pentru a elimina din memoria internă a calculatorului programul existent. De asemenea, dacă doriți, puteți introduce pe o linie program mai multe instrucțiuni separate prin simbolul " : ".



Aplicație. Introduceți și executați următoarele programe BASIC-80.

a)
```
10 PRINT "EXEMPLU DE ADU
   NARE"
20 AANR = 8
30 ABNR = 5
40 ACNR = AANR + ABNR
50 PRINT : PRINT
60 PRINT AANR " + " ABNR
   " = " ACNR
70 END
```

b)
```
10 PRINT "EXEMPLU DE SCĂDERE"
20 AANR = 75
30 ABNR = 100
40 ACNR = AANR - ABNR
50 PRINT : PRINT : PRINT
60 PRINT AANR " - " ABNR
   " = " ACNR
70 END
```

c)
```
10 PRINT "EXEMPLU DE ÎNMULŢIRE"
20 AANR = 12
30 ABNR = 13
40 ACNR = AANR * ABNR
50 PRINT : PRINT
60 PRINT AANR ." * " ABNR
   " = " ACNR ·
70 END
```

d)
```
10 PRINT "EXEMPLU DE ÎMPĂRŢIRE"
20 AANR = 120.20
30 ABNR = 195
40 ACNR = AANR / ABNR
50 PRINT : PRINT
60 PRINT AANR " / " ABNR " = "
   ACNR
70 END
```

e)
```
10 PRINT "EXEMPLU DE RIDICARE LA
   PUTERE"
20 AANR = 13
30 ABNR = 2 ↑ AANR
40 PRINT "2 LA PUTEREA" AANR " = "
   ABNR
50 END
```

f)
```
10 A = 95630
20 B = 8523.
30 C = A * B
40 PRINT A, " * ", B, " = ", C
50 PRINT A " * " B " = " C
60 PRINT A; *; B; =; C
70 END
```

□ Particularităţi ale programării în limbajul BASIC-PLUS

vol. 2, pag. 200

Priviţi programul. Nu vă grăbiţi în parcurgerea lui.

```
10 R=3
20 A=4 * PI * R ** 2
30 PRINT A
40 PRINT "A=", A
50 PRINT "A="; A
60 PRINT 4; " * "; PI; " * "; R; " ↑ "; 2; "="; A
70 PRINT 4; " * "; PI; " * "; R; " ↑ "; 2; "="; 4 * PI * R ↑ 2
```

Analizaţi fiecare instrucţiune în parte şi comparaţi-o cu cea din programele precedente. Aţi descoperit vreo deosebire? Dacă aţi găsit-o înseamnă că aţi făcut o analiză riguroasă, dacă nu, reluaţi lecturarea programului cu mai multă atenţie, după cum urmează:

10 Atribuie variabilei R valoarea 3.

20 Atribuie valoarea expresiei numerice $4\pi R^2$ variabilei A. Operatorul aritmetic pentru ridicarea la putere este " **".

30 Tipăreşte valoarea calculată în linia precedentă.

40 Virgula decalează imprimarea valorii variabilei A la începutul zonei următoare. Linia terminal se consideră împărţită în zone care au o lungime de 14 caractere.

50 Separatorul punct şi virgulă determină imprimarea valorii lui A la două spaţii după semnul egal (valoarea extrasă este numerică pozitivă).

60, 70 Editează valoarea ariei rezervorului (A), utilizîndu-se două moduri diferite de calcul.

Remarci

- Numele variabilelor simple sînt formate dintr-o literă sau o literă urmată de o singură cifră.
- Variabilele pot fi numerice şi variabile şir.

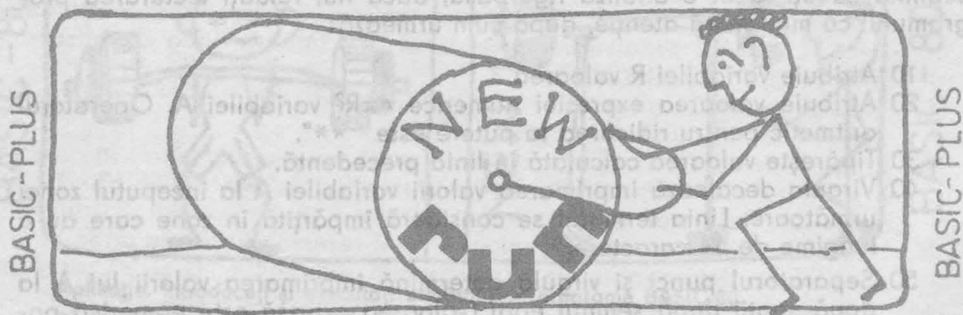
- Variabilele simple numerice pot fi întregi (referă valori numerice de tip întreg ce se reprezintă pe un cuvînt) și reale (referă valori numerice de tip real ce se reprezintă pe două cuvinte).
- Tipul întreg al unei variabile se indică prin caracterul "%" adăugat numelui acesteia.
- Într-un program BASIC-PLUS pot exista două variabile cu același nume dar de tipuri diferite.
- Într-un program pot exista variabile simple numerice și variabile simple șir cu același nume.
- Sînt permise două tipuri de constante: numerice și șiruri.
- BASIC-PLUS recunoaște constantele numerice întregi și constantele reale.
- Constantele numerice pot fi reprezentate în format: întreg, zecimal, exponențial.
- Operațiile aritmetice admise sînt: adunarea (+), scăderea (−), înmulțirea (*), împărțirea (/), ridicarea la putere (**).
- Dacă într-o expresie aritmetică se folosesc variabile cărora nu li s-au atribuit anterior o valoare, expresia rămîne nedefinită.
- Prioritatea operațiilor este cea cunoscută.
- Structura generală a unei instrucțiuni de atribuire cuprinde cuvîntul opțional **LET** urmat, în mod obligatoriu, de numele uneia sau mai multor variabile (atribuire multiplă), de semnul de egalitate și de o expresie numerică.
- Dacă se atribuie unei variabile numerice o constantă de un tip diferit de al variabilei, atunci valoarea numerică se convertește automat la tipul variabilei (vezi linia 10).
- La evaluarea expresiilor, toți operanzii sînt convertiți automat la tipul operandului de precizie maximă care este și tipul rezultatului expresiei. Astfel, dacă un operand este de tip întreg iar celălalt este de tip real se face mai întîi conversia tipului întreg în tip real după care se execută operația cu valori de tip real.
- BASIC-PLUS acceptă linii multiinstrucțiune.

TEST

Precizați valoarea lui A în urma execuției următorului program:

```
10 PRINT
20 A = 4 * PI * R ** 2
30 PRINT A
```

R. Zero, deoarece variabilei R nu i s-a atribuit nici o valoare.



Dacă doriți să introduceți programul în calculatoarele INDEPENDENT, CORAL dați mai întîi comanda **NEW** după care apăsați pe **[RETURN]**. (v. fig. 2.4). Tastați **RUN, [RETURN]** și veți obține, în urma execuției programului, în diferite formate de editare 113.097 ce reprezintă aria rezervorului de rază 3 m.

a)

```
10 R = 3
20A = 4 * PI * R * * 2
30PRINT A
1000END
:RUN
 113.097
END OF PROGRAM AT LINE 1000
```

b)

```
10R = 3
20A = 4 * PI * R * * 2
50 PRINT  A = ";A
1000 END

:RUN
A = 113.097
END   OF PROGRAM AT LINE 1000
```

c)

```
10R = 3
20A = 4 * PI * R * * 2
40 PRINT "A=", A
1000 END
:RUN
A =                    113.097
END OF PROGRAM AT LINE 1000
```

d)

```
10 R = 3
20A = 4*PI*R**2
60 PRINT 4;"X";PI;"X";R;"^";2;"=";A
70 PRINT 4;"*";PI;"*";R;"↑2";2 =";4*PI*R↑2
1000 END
:RUN
4×3.14159×3^2 = 113.097
4*3.14159*3↑2 = 113.097
END OF PROGRAM AT LINE 1000
```

Fig. 2.4. Rezultatele execuției *(a, b, c, d)* programului.

Aplicaţie. Într-un triunghi dreptunghic isoscel se cunoaşte lungimea catetei, A1=A2=8 cm. Să se scrie un program BASIC pentru calculul ariei triunghiului (S) şi înălţimii (H) duse din vîrful unghiului drept.

```
10 PRINT                    50 B = (2 * A1 ** 2) ** 0.5
20 A1, A2 = 8               60 H = A1 * A1 / B
30 S = (A1 * A2) / 2        70 PRINT " AD = ", H
40 PRINT "ARIA =", S        80 END
```

TESTE

- Să se calculeze debitul de volum al apei ce curge printr-o conductă orizontală cu secţiunea transversală variabilă. Se cunosc: secţiunile transversale $S1=10^{-3}$ m², $S2=2\cdot10^{-3}$ m², şi diferenţa de nivel dintre coloanele de lichid din sondele de presiune pe conductă $\Delta h=0{,}2$ m. Se va utiliza relaţia de calcul:

$$Q_v=S_1S_2\sqrt{2g\Delta h(S_2^2-S_1^2)} \quad (m^3/s).$$

- Un corp cu masa m=2 kg este aruncat orizontal cu viteza iniţială $v_0=20$ m/s. Să se calculeze energia cinetică a corpului după un interval de timp $\Delta t=4$ s. Se va utiliza relaţia de calcul $E_0=\frac{mv^2}{2}=\frac{m}{2}\cdot(v_0^2+g^2\Delta t^2)$, unde $g=9{,}81$ m/s².
- Să se calculeze rezultanta a două forţe de mărime egală $F=\sqrt{2}$ N avînd direcţii perpendiculare.
- Un corp cu masa m=8 kg se deplasează fără frecare pe un plan orizontal cu acceleraţia a=0,5 m/s². Să se calculeze: a) mărimea forţei orizontale care deplasează corpul; b) viteza corpului la momentul t=30 s; c) distanţa parcursă de corp în timpul t=30 s; d) energia cinetică a corpului în momentul t=20 s. Se vor utiliza formulele: $F=ma$ (N); $v=at$ (m/s); $x=at^2/2$ (m); $E_0=\frac{ma^2t^2}{2}$ (J).
- Să se calculeze volumul corpului obţinut prin rotaţia unei suprafeţe hexagonale regulate în jurul unei laturi, ştiind că latura hexagonului (a) este egală cu 8 cm. Se va utiliza relaţia: $V=\frac{9\pi a^3}{2}$.

□ Particularităţi ale programării în limbajul ABASIC

vol. 2, pag. 201

Acum, după ce aveţi experienţa limbajului BASIC implementat pe calculatoare personale, micro şi mini calculatoare, ce puteţi spune despre acest program? Identificaţi pentru început cele două variabile A şi R. Ele sînt variabile numerice reale. În ABASIC variabilele numerice pot fi: numerice reale şi numerice întregi. O variabilă este identificată printr-un şir de caractere ce începe cu o literă urmată de un număr oarecare de litere sau cifre. În vederea precizării explicite a tipului, ea se termină, opţional, cu unul din caracterele "!"; "#" pentru variabile numerice reale şi cu "%" pentru variabile numerice întregi.

Variabilele care nu au precizat un tip implicit sînt împărţite în clase de tipuri, funcţie de primul caracter al numelui (prin instrucţiuni ca: **DEFINT, DEFDBL**). Dacă precizarea tipurilor implicite nu este făcută de utilizator, atunci toate variabilele sînt considerate numerice reale.

ABASIC permite ca **operaţii aritmetice:** adunarea (+), scăderea (–), înmulţirea (*), împărţirea (/), ridicarea la putere (** sau ↑), împărţirea întreagă (\, // sau **DIV**) şi restul împărţirii întregi **(MOD)**.

Instrucțiunea **LET** (opțională) permite și o atribuire multiplă. Dacă este cazul se realizează trunchierea valorii reale la întreg.

Cît privește instrucțiunea **PRINT** menționăm că în limbajul ABASIC linia terminal se consideră împărțită pe zone de lungime 14 caractere. Pentru a defini alte valori pentru lungimea liniei de imprimat (implicit 132) și lungimea unei zone (implicit 14), utilizatorul dispune de instrucțiunea **OPTION WIDTH** al cărei format este:

Format general
OPTION WIDTH <expresie 1>, <expresie 2>

unde valorile convertite la întregi pentru ⟨expresie 1⟩ și ⟨expresie 2⟩ reprezintă lungimea liniei, respectiv a zonei de imprimat.

Semnificația separatorilor "," și ";" este aceeași ca la BASIC-80 și BASIC-PLUS.

ABASIC NEW NEW ABASIC

În sfîrșit, o ultimă remarcă privind ștergerea programelor. Un program încărcat poate fi șters în totalitate prin execuția instrucțiunii **NEW**.

Și-acum nu vă mai rămîne decît să introduceți programul, să-l executați și să vă convingeți singuri de cele afirmate anterior.

TEMA 2

☐ Răspundeți prin DA sau NU la următoarele întrebări:

- A reprezintă un nume de variabilă numerică acceptată de toate versiunile BASIC studiate.
- LET poate fi un nume de variabilă pentru BASIC-80.
- AB poate fi un nume de variabilă pentru BASIC-COMMODORE.
- Cuvîntul rezervat **LET** este obligatoriu pentru o instrucțiune de atribuire în BASIC HC-85, TIM S, SPECTRUM.
- Toate calculatoarele personale pot realiza operații matematice.
- Pentru calculatoarele aMIC, INDEPENDENT, CORAL numele unei variabile numerice este format dintr-o literă urmată de cel mult o cifră.

– Într-o variabilă BASIC-80, COMMODORE numai primele două caractere sînt semnificative.
– Caracterele "%", "!", "#" pot fi ataşate unei variabile BASIC-80 şi ABASIC.
– Caracterul "%" ataşat unei variabile BASIC-80, ABASIC, BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-PLUS precizează tipul întreg al unei variabile.
– Caracterul "#" poate fi ataşat unei variabile BASIC-80, ABASIC dar nu poate fi ataşat unei variabile BASIC-aMIC, GWBASIC.
– Rezultatul execuţiei următorului program este zero.

```
10 Y = 5
20 PRINT X + Y – 5
```

– Rezultatul execuţiei următorului program este 8.

```
10 I = 4
20 I = + 4
30 PRINT I
```

– Rezultatul execuţiei următorului program este 8.

```
10 W = 7
20 C = W + 1
30 PRINT 7
```

□ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Într-o variabilă BASIC .... numai primele două caractere sînt semnificative, pe cînd într-o variabilă BASIC .... numai primele 40 caractere sînt semnificative.

b) Pentru ştergerea programului curent din memoria calculatorului .... se utilizează comanda **SCR**.

c) Comanda **NEW** nu se utilizează pe calculatoarele ...., ..., ..., în timp ce comanda **SCR** este folosită doar pentru calculatorul ....

d) Separatorul apostrof este specific limbajului BASIC .... şi face ca valorile să fie tipărite ....

e) Instrucţiunea **PRINT** poate fi înlocuită cu .... în limbajul BASIC implementat pe calculatoarele ...., ...

f) Liniile multiinstrucţiune nu sînt admise în limbajul BASIC implementat pe calculatorul ....

□ Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Fiind date coordonatele punctelor M(3, 4) şi N(–3.4, 5.75), să se calculeze şi afişeze valoarea distanţei MP=PN, unde P(X, Y) este mijlocul segmentului MN.

b) Să se calculeze şi afişeze coordonatele punctului de intersecţie al dreptelor de ecuaţii:

$$Y=3x+7$$

şi

$$Y=-4x+14$$

Observaţie. Dreptele se intersectează deoarece pantele lor (3 şi 4) sînt diferite între ele.

c) Relaţia pentru calculul inductanţei unei bobine este $L=\frac{\mu_0\mu_r N^2S}{l}$ în care μ_0 reprezintă permeabilitatea vidului, μ_r – permeabilitatea relativă, N – numărul de spire, S – aria secţiunii, l – lungimea bobinei.

Să se calculeze inductanţa unei bobine cunoscîndu-se: $\mu_0=4\pi 10^{-7}$ N/A^2, $\mu_r=300$, N=1 000, S=10^4 cm^2, l=62,8 cm.

Observaţie. Valorile lui S şi l se vor exprima în SI (sistemul internaţional), respectiv m^2 şi m.

d) Randamentul motorului Otto se exprimă prin relaţia $\eta=1-\frac{1}{\varepsilon^{\gamma-1}}$ unde η reprezintă randamentul, $\varepsilon=\frac{V_1}{V_2}$ reprezintă raportul de compresie, iar γ reprezintă exponentul adiabatic.

Să se calculeze randamentul unui motor Otto, dacă se cunoaşte $\varepsilon=5$ şi $\gamma=1,4$ (substanţa de lucru este aerul).

e) Să se calculeze randamentul motorului DIESEL cunoscînd $\varepsilon=10$, $\rho=2$, $\gamma=1,4$. Se va utiliza relaţia:

$$\eta=1-\frac{\rho^\gamma-1}{\gamma\varepsilon^{\gamma-1}(\rho-1)}$$

f) Să se calculeze constanta universală a gazelor (R) cunoscîndu-se: $p_0=101\,325$ N/m^2, $V\mu_0=22,42$ m^3/kmol, $T_0=273,15$ K. Se va utiliza relaţia:

$$R=\frac{p_0V\mu_0}{T_0}.$$

g) Să se afle volumul cilindrului a cărui secţiune axială este un pătrat de arie a=16 m^2. Se va utiliza relaţia $V=\pi R^2h=\frac{a\sqrt{a}}{4}\pi$.

h) Să se calculeze volumul conului înscris într-un tetraedru regulat de muchie a=10 m. Se va utiliza relaţia:

$$V=\frac{\pi a^3\sqrt{6}}{108}\left(R=\frac{a\sqrt{3}}{6},\ h=\frac{a\sqrt{3}}{6}\right)$$

i) Să se scrie ecuaţia perpendicularei care trece prin mijlocul dreptei determinată de punctele A(1, 2) şi B(4, 7).

Observaţie. Diferenţele x_2-x_1 şi y_2-y_1 sînt diferite de zero (3 respectiv 5).

☐ **Să se scrie un program BASIC care calculează şi afişează** cantităţile totale trimestriale de jucării fabricate de întreprinderea „URSU-

LEȚUL", cunoscînd cantitățile de jucării (în bucăți) fabricate în fiecare lună: ianuarie (3 000), februarie (8 000), martie (7 500), aprilie (6 000), mai (5 000), iunie (2 300), iulie (1 000), august (3 200), septembrie (400), octombrie (600), noiembrie (400), decembrie (9 000).

☐ **Să se scrie un program BASIC care calculează și afișează** consumul mediu anual al unui articol cunoscînd consumurile anuale (în bucăți) pe ultimii patru ani: 1984 (321), 1985 (432), 1986 (766), 1987 (876).

SOLUȚIA TEMEI 2

☐ **Răspunsurile sînt:**

DA, NU, DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, NU.

☐ **Cuvintele lipsă sînt:**

a) PRAE, COMMODORE; 80; **b)** aMIC; **c)** aMIC, AMSTRAD; **d)** HC-85, TIM S, SPECTRUM; una sub alta; **e)** COMMODORE, AMSTRAD, FELIX C; **f)** aMIC.

☐ **Programele BASIC sînt:**

a) Problema se reduce la calculul coordonatelor x și y ale punctului P. Deoarece punctul P se găsește la mijlocul distanței MN, putem scrie că:

$$x=\frac{x_1+x_2}{2},$$

$$y=\frac{y_1+y_2}{2}.$$

În program se vor atribui variabilelor x1, y1 coordonatele punctului M, iar variabilelor x2, y2 coordonatele punctului N. În final se va calcula distanța MP cu formula cunoscută (v. și aplicația de la pag. 181), după care se va tipări valoarea calculată.

Prezentăm în continuare lista programului rulat pe calculatorul personal FELIX PC.

```
NEW
20 y1 = 4
10 x1 = 3
30 x2 = - 3 . 4
35 x = (x1 + x2)/2
40 y2 = 5 . 75
50 MP = (x2 - x) ↑ 2 + (y2 - y) ↑ 2) ↑ 0 . 5
60 PRINT "DISTANȚA MP = ", MP
45 y = (y1 + y2)/2
```

b)
```
10 LET M1 = 3
20 LET N1 = 7
30 LET M2 = - 4
40 LET N2 = 14
50 PRINT " Y = ( "; M1; " ) *
   X + ( "; N1; " ) "
60 PRINT " Y = ( "; M2; " ) *
   X + ( "; N2; " ) "
70 LET X = (N2 - N1) / (M1 - M2)
80 LET Y = M1 * X + N1
90 PRINT
100 PRINT "PUNCTUL DE INTERSECȚIE
    (";X;", ";Y;")"
110 END
```

Semnificația instrucțiunilor este următoarea:

10 Atribuie variabilei M1 panta primei drepte.
20 Atribuie variabilei N1 valoarea termenului liber al ecuației primei drepte.
30 Atribuie variabilei M2 panta celei de-a doua drepte.
40 Atribuie variabilei N2 valoarea termenului liber al ecuației celei de-a doua drepte.
50 Tipărește ecuația primei drepte Y=(3) * X+(7).
60 Tipărește ecuația celei de-a doua drepte Y=(−4) * X+(14).
70 Calculează abscisa punctului de intersecție.
80 Calculează ordonata punctului de intersecție.
90 Tipărește un rînd liber.
100 Editează rezultatul final.

Remarcă. Inainte de tastarea acestui program, dați comenzile **SCR** (aMIC), și **NEW** (PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE, M-118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX-C) pentru ștergerea programului curent din memoria calculatorului. Utilizarea cuvîntului **LET** în instrucțiunile 10, 20, 30, 40 70, 80 este obligatorie numai pentru calculatoarele HC-85, TIM S, SPECTRUM. Executați programul cu comanda **RUN.**

ABASIC SCR RUN ABASIC

c)
```
10 PRINT "INDUCTANTA BOBI-
   NEI = "; (4 * PI / 10 ↑ 7 *
   300 * 1000 ↑ 2) / (62 . 8 / 100); "H"
20 END
```

d)
```
10 PRINT "RANDAMENTUL = ";
   1 - 1/ 5 ↑ (1 . 4 - 1)
20 END
```

e)
```
10 LET D = 2
20 LET E = 10
30 LET G = 1 . 4
40 LET R = 1 - (D ↑ G - 1) /
   (G * E ↑ (G - 1) * (D - 1))
50 PRINT "RANDAMENTUL = ";
   R
```

f)
```
10 PRINT "R = "; 101325 *
   22 . 42 / 273 . 15
```

g)
```
10 PRINT "VOLUMUL = ";
   16 * 16 ↑ 0 . 5 * PI / 4
20 END
```

h)
```
10 LET A = 10
20 LET V = PI * A ↑ 3 * 6 ↑ 0 . 5 / 108
30 PRINT "VOLUMUL = "; V
40 END
```

i)
```
10 LET X1 = 1
20 LET Y1 = 2
30 LET X2 = 4
40 LET Y2 = 7
50 PRINT " ( "; X1; ", "; Y1; ") SI
   ( "; X2; ", "; Y2; " ) "
60 PRINT "ECUATIA ESTE: ";
70 LET X0 = (X1 + X2)/2
```

```
80 LET Y0 = (Y1 + Y2) / 2
90 LET M = (Y2 – Y1)/(X2 – X1)
100 LET M0 = – 1 / M
110 LET N = Y0 – M0 * X0
120 PRINT "Y = "; M0; " * X +
    ("; N; " ) "
130 END
```

□ **Programul pentru calculul cantităților totale de jucării fabricate este:**

```
10 LET I1 = 3000
20 LET F = 8000
30 LET M1 = 7500
40 LET A1 = 6000
50 LET M2 = 5000
60 LET I2 = 2300
70 LET I3 = 1000
80 LET A2 = 3200
90 LET S = 400
100 LET I4 = 600
110 LET N = 400
120 LET D = 9000
130 LET T1 = I1 + F + M1
140 LET T2 = A1 + M2 + I2
150 LET T3 = I3 + A2 + S
160 LET T4 = I4 + N + D
170 PRINT
180 PRINT "TOTAL JUCARII TRIMESTRUL
    1 = "; T1
190 PRINT
200 PRINT "TOTAL JUCARII TRIMESTRUL
    2 = "; T2
210 PRINT
220 PRINT "TOTAL JUCARII TRIMESTRUL
    3 = "; T3
230 PRINT
240 PRINT "TOTAL JUCARII TRIMESTRUL
    4 = "; T4
250 END
```

□ **Programul pentru calculul consumului mediu anual este:**

```
10 LET A1 = 321
20 LET A2 = 432
30 LET A3 = 766
40 LET A4 = 876
50 LET C = (A1 + A2 + A3 + A4) / 4
60 PRINT "CONSUMUL MEDIU ANUAL = "; C
70 END
```

CONVERSAȚIA 3

Proiectarea și realizarea unui program pentru calculul ariei și volumului unui rezervor sferic a cărui rază poate lua diferite valori. Metodică de analiză, proiectare și realizare a programelor BASIC. Introducerea dinamică a datelor. Instrucțiunea INPUT. Documentarea programelor. Instrucțiunea REM. Salvarea unui program BASIC pe medii magnetice. Depanarea unui program. APLICAȚII și TESTE pentru cititor



EXEMPLELE 3 – PC, m, M, F

□ Metodică de analiză, proiectare şi realizare a programelor

Formularea problemei

Vom relua în cadrul acestei conversaţii problema rezervorului sferic, complicînd-o sub următorul aspect: nu mai stabilim de la început valoarea razei (R=3), ci o vom transmite calculatorului la solicitarea acestuia. De asemenea, ne propunem să calculăm şi volumul rezervorului pe care să-l afişăm, alături de valorile razei şi ariei (calculate).

Ce problemă simplă, veţi exclama! Într-adevăr, problema nu este dificilă, dar trebuie să recunoaştem că ceea ce pare simplu pentru unii este foarte complicat pentru alţii.

Desigur, ea poate fi rezolvată foarte bine şi de către un elev de şcoală elementară sau chiar de un adult (părinţii elevului!). Obiectivul nostru, însă, este rezolvarea informatică a problemei cu ajutorul calculatorului personal, microcalculatorului etc.

Înainte de a scrie un program trebuie să cunoaştem şi să înţelegem foarte bine problema. În cazul nostru, problema este una dintre cele mai simple, calculul ariei şi volumului unui rezervor sferic atunci cînd se cunoaşte raza.

Procesul de alcătuire a programului, pe care îl vom urma pe tot parcursul acestei lucrări*, constă din următoarele faze: analiză, proiectare, codificare (implementare) în limbajul BASIC (pe calculatoare personale, microcalculatoare, mini şi maxi calculatoare), testare şi depanare.

Faza de analiză

În cadrul acestei faze se poartă o discuţie cu persoana care a solicitat realizarea programului, urmînd să aflăm cum doreşte să-i fie prezentată lista cu rezultate, care sînt datele de intrare, precum şi intenţiile programului solicitat. În rezolvarea acestei probleme se urmăreşte ca pe ecranul monitorului să se afişeze simplu, într-o formă deloc pretenţioasă, cele trei valori pentru: rază, arie şi volum (vezi modulul de analiză structurată, fig. 3.1 (a, b).

În rezolvarea informatică a unei probleme, întotdeauna se porneşte cu datele de ieşire, urmînd să ne întrebăm apoi dacă avem suficiente date de intrare pentru a putea fi obţinute ieşirile dorite. În cazul nostru răspunsul este afirmativ.

Dacă răspunsul la această întrebare este negativ, vom defini încă un tip de date şi anume **datele de stare,** care nu sînt nici de intrare şi nici de ieşire. Asociind fiecărui tip de date cîte un nume de **variabilă** obţinem **variabilele de intrare, de ieşire** şi **de stare** care, structurate într-o tabelă, alcătuiesc ceea ce numim **tabela de variabile.**

---

* Vezi şi „Învăţăm FORTRAN... conversînd cu calculatorul", „Învăţăm COBOL... conversînd cu calculatorul".

R = A = V =

a)

RAZA (NR. REAL) RAZA (NR. INTREG)
XX.XX XX

R = XX.XX A = XXX.XX V = XXX.XX

b)

TABELA DE VARIABILE

Nume program: EXEMPLELE 3–PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieşire
R: raza rezervorului		A: aria rezervorului V: volumul rezervorului

c)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 3–PC, m, M, F

Descrierea programului

Programul citeşte de la terminal valoarea razei (pe care o şi tipăreşte), calculează, afişează aria şi volumul rezervorului sferic.

Intrări

Raza rezervorului.

Ieşiri

Lista cu raza, aria şi volumul rezervorului sferic.

Lista de funcţiuni ale programului

1. Citire rază
2. Calculul ariei rezervorului (sferic)
3. Calculul volumului rezervorului (sferic)
4. Afişarea rezultatului
5. Stop

d)

Fig. 3.1, *a–e*. Modulul de analiză structurată; *a, b)* formatul datelor de ieşire; *c)* tabela de variabile; *d)* specificaţiile de programare;

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE	
Nume program: EXEMPLELE 3–PC, m, M, F	
Modul	**Funcţiuni**
CIT-R	1
PREL-R	2,3
REZULTATE	4,5

Fig. 3.1. *e)* alocarea funcţiunilor de prelucrare.

Tabela de variabile pentru EXEMPLELE 3 – PC, m, M, F este reprezentată în modulul de analiză structurată, fig. 3.1 (c).

Specificaţii de programare

Înainte de a începe proiectarea unui program, trebuie sa avem o descriere generală a acestuia, o identificare a intrărilor şi a ieşirilor, precum şi o listă a funcţiunilor (paşilor) programului. Împreună, aceste elemente de documentare reprezintă specificaţiile de programare. Ele sînt scrise în general de către un grup de analişti sau programatori, respectînd o anumită metodologie. Specificaţiile de programare pentru EXEMPLELE 3 – PC, m, M, F sînt prezentate în modulul de analiză structurată, fig. 3.1 (d).

Remarcă. Este important ca schemele şi specificaţiile de programare să fie validate şi de persoana care a solicitat realizarea programului. Această aprobare ajută la asigurarea interpretării corecte a programului dorit. Dacă sînt erori, ele pot fi înlăturate înainte de a se fi depus un volum de muncă pentru proiectarea programului.

Faza de proiectare

Asigurîndu-ne că nu s-au strecurat greşeli în faza de analiză a problemei, putem aborda în continuare faza de proiectare a programului, utilizînd ca instrumente de proiectare: diagrama de structură, pseudocodul şi schema logică.

Diagrama de structură

Diagrama de structură (similară clasicei organigrame a unei întreprinderi) reflectă, într-un mod simplu de reprezentare grafică, structura logică de prelucrare a unui program, pe nivele de detaliere, conform următoarelor **reguli.**

- Nivelele de detaliere se parcurg de sus în jos (TOP DOWN);
- Fiecare nivel reprezintă o detaliere a nivelului precedent;
- Blocurile situate pe un acelaşi nivel se parcurg de la stînga la dreapta;
- Terminarea parcurgerii unui nivel presupune parcurgerea următorului bloc din nivelul anterior.

Într-o diagramă de structură blocurile se reprezintă (grafic) printr-un dreptunghi în interiorul căruia se trece numele modulelor de prelucrare (vezi modulul de proiectare structurată, fig. 3.2 (a).

Funcţiunile modulelor de prelucrare (CIT-R, PREL-R, REZULTATE) corespund listei de funcţiuni ale programului (faza de analiză) din cadrul specificaţiilor de programare (vezi modulul de analiză structurată, fig. 3.1 (e).

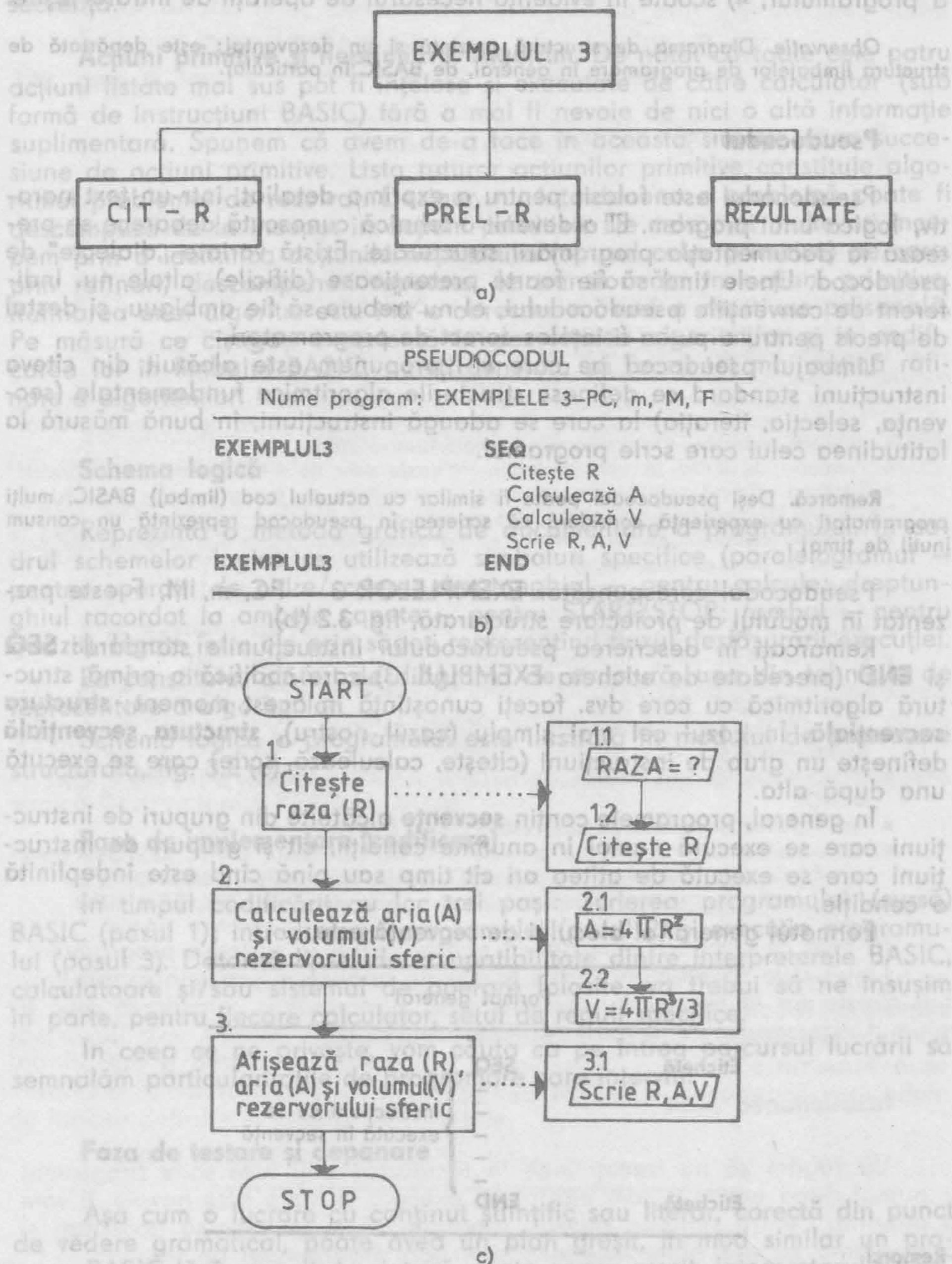


Fig. 3.2. Modulul de proiectare structurată: *a)* diagrama de structură; *b)* pseudocodul; *c)* schema logică.

Dintre virtuţile diagramei de structură amintim următoarele: 1) este independentă de limbaj sau tip calculator; 2) poate reda o structură a datelor sau a logicii de prelucrare; 3) scoate în evidenţă structura funcţională a programului; 4) scoate în evidenţă necesarul de operaţii de intrare/ieşire.

Observaţie. Diagrama de structură prezintă şi un dezavantaj: este depărtată de structura limbajelor de programare în general, de BASIC în particular.

Pseudocodul

Pseudocodul este folosit pentru a exprima detaliat, într-un text narativ, logica unui program. El a devenit o tehnică cunoscută deoarece se pretează la documentaţia programării structurate. Există variate „dialecte" de pseudocod. Unele tind să fie foarte pretenţioase (dificile), altele nu. Indiferent de convenţiile pseudocodului, el nu trebuie să fie ambiguu, ci destul de precis pentru a putea fi înţeles corect de programatori.

Limbajul pseudocod pe care vi-l propunem este alcătuit din cîteva instrucţiuni standard ce definesc structurile algoritmice fundamentale (secvenţa, selecţia, iteraţia) la care se adaugă instrucţiuni, în bună măsură la latitudinea celui care scrie programul.

Remarcă. Deşi pseudocodul poate fi similar cu actualul cod (limbaj) BASIC, mulţi programatori cu experienţă consideră că scrierea în pseudocod reprezintă un consum inutil de timp!

Pseudocodul corespunzător EXEMPLELOR 3 – PC, m, M, F este prezentat în modulul de proiectare structurată, fig. 3.2 (b).

Remarcaţi în descrierea pseudocodului instrucţiunile standard: **SEQ** şi **END** (precedate de eticheta EXEMPLUL 3) care codifică o primă structură algoritmică cu care dvs. faceţi cunoştinţă în acest moment: **structura secvenţială.** În cazul cel mai simplu (cazul nostru), **structura secvenţială** defineşte un grup de instrucţiuni (citeşte, calculează, scrie) care se execută una după alta.

În general, programele conţin **secvenţe** alcătuite din grupuri de instrucţiuni care se execută numai în anumite condiţii, cît şi grupuri de instrucţiuni care se execută de atîtea ori cît timp sau pînă cînd este îndeplinită o condiţie.

Formatul general al blocului de secvenţă este:

Format general		
Etichetă	**SEQ**	
	–	Instrucţiuni ce se
	–	execută în secvenţă
	–	
	–	
Etichetă	**END**	

Remarci

- Instrucţiunile **SEQ** şi **END** au obligatoriu aceeaşi etichetă;
- Instrucţiunea **SEQ** reprezintă punctul unic de intrare în bloc;
- Instrucţiunea **END** reprezintă punctul unic de ieşire din bloc.

Particularizînd formatul general al blocului de secvenţă la EXEMPLELE 3 – PC, m, M, F, observaţi cum EXEMPLUL 3 joacă rolul etichetei (aceeaşi pentru instrucţiunile **SEQ** şi **END**, iar: citeşte R, calculează A, calculează V, scrie R, A, V reprezintă acţiuni (instrucţiuni) ce se execută în secvenţă.

Acţiuni primitive şi neprimitive. Algoritm. De notat că toate cele patru acţiuni listate mai sus pot fi înţelese şi executate de către calculator (sub formă de instrucţiuni BASIC) fără a mai fi nevoie de nici o altă informaţie suplimentară. Spunem că avem de-a face în această situaţie cu o succesiune de acţiuni primitive. Lista tuturor acţiunilor primitive constituie algoritmul problemei de rezolvat. Desigur, nu întotdeauna o problemă poate fi descompusă de la început în acţiuni primitive. De cele mai multe ori începem prin a identifica acţiunile neprimitive (opusul celor primitive) pe care, prin rafinări, descompuneri repetate, le transformăm în acţiuni primitive. Rafinarea unui algoritm este într-o oarecare măsură o chestiune personală. Pe măsură ce cîştigaţi experienţă în dezvoltarea algoritmilor şi în codificarea lor în limbajul BASIC, puteţi constata că faceţi tot mai puţină rafinare a algoritmilor.

Schema logică

Reprezintă o metodă grafică de documentare a programului. În cadrul schemelor logice se utilizează simboluri specifice (paralelogramul – pentru operaţii de citire/scriere; dreptunghiul – pentru calcule; dreptunghiul racordat la ambele capete – pentru START/STOP; rombul – pentru decizii), legate între ele prin săgeţi reprezentînd fluxul desfăşurării execuţiei.

Ea constituie, alături de diagrama de structură, una din tehnicile de reprezentare a algoritmilor.

Schema logică a programului este ilustrată în modulul de proiectare structurată, fig. 3.2 (c).

Faza de implementare (codificare)

În timpul codificării au loc trei paşi: scrierea programului (sursă) BASIC (pasul 1); introducerea programului (pasul 2) şi execuţia programului (pasul 3). Datorită lipsei de compatibilitate dintre interpreterele BASIC, calculatoare şi/sau sistemul de operare folosite, va trebui să ne însuşim în parte, pentru fiecare calculator, setul de reguli specifice.

În ceea ce ne priveşte, vom căuta ca pe întreg parcursul lucrării să semnalăm particularităţile de programare care intervin.

Faza de testare şi depanare

Aşa cum o lucrare cu conţinut ştiinţific sau literar, corectă din punct de vedere gramatical, poate avea un plan greşit, în mod similar un program BASIC fără greşeli de sintaxă poate opera greşit. Interpretorul BASIC nu dispune de nici o metodă pentru a cunoaşte ceea ce doreşte progra-

matorul să obţină prin program. În faza de testare, programatorul se asigură dacă programul realizat este în concordanţă cu specificaţiile de programare.

☐ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 201

Scrierea programului (pasul 1). Reguli generale

Limbajul BASIC-aMIC dispune de cîteva reguli generale de scriere (sintaxă) ce trebuie cunoscute şi respectate cu stricteţe.

REGULI

- Pentru scrierea unui program BASIC se folosesc numai litere mari.
- Fiecare linie program începe cu un număr de linie.
- Numărul de linie este un întreg cuprins între 1 şi 32767.
- Liniile program conţin o singură instrucţiune **(linie simplă).**
- Numele variabilelor simple sînt formate dintr-o literă urmată de cel mult o cifră.
- Numele variabilelor indexate sînt formate dintr-o literă.
- Numărul de indici ai unei variabile indexate este de 1–2 (vector sau matrice).
- Sistemul elimină spaţiile din orice şir de caractere, cu excepţia celor din textele şirurilor de caractere (cuprinse între ghilimele).
- În aMIC instrucţiunea **END** trebuie să fie ultima instrucţiune din program (cu numărul de linie cel mai mare).
- Operatorii aritmetici admişi sînt: + (adunare); – (scădere); * (înmulţire); / (împărţire); ↑ (ridicare la putere).
- Operatorii de relaţie admişi sînt: =; <; >; <=; >=; <> (diferit).
- Limbajul BASIC-aMIC nu admite operatori logici **(NOT, AND, OR).**
- Funcţiile matematice standard admise sînt: **PI, SIN, COS, TAN, ATN, LOG, EXP, SQR, ABS, INT, SGN, RND.**
- Funcţiile admise pentru lucrul cu şirurile de caracetre sînt: **VAL, STR$, LEN, CHR$, INKEY$.**
- Instrucţiunile de salt recunoscute de interpretor sînt: **GOTO, IF... THEN, ON... GOTO.**
- Instrucţiunea de ciclare permisă este **FOR... NEXT.**
- Instrucţiunile de intrare/ieşire permise sînt: **INPUT, READ, DATA, RESTORE** (fără număr de linie), **PRINT, PRINT AT, MAT READ, MAT INPUT, MAT PRINT.**
- Alte funcţii (pentru prelucrarea matricilor) admise sînt: **TRN, ZER, CON, IDN.**

În cele ce urmează vom aplica aceste reguli la scrierea programului. După cum vă reamintiţi (vezi fazele de analiză şi proiectare), programul citeşte de la tastatură valoarea razei rezervorului, calculează aria şi volumul rezervorului iar în final tipăreşte rezultatele.

Instrucţiunea REM

Vă rugăm să nu treceţi încă la tastatură! Întîi vom scrie programul şi numai după aceea îl vom introduce, executa şi dacă este nevoie, îl vom şi depana. Pentru scrierea acestui program, vă propunem să acceptaţi şi colaborarea cu noi! Vom începe prin a „da programului un titlu".

10 **REM** EXEMPLUL 3 – PC (AMIC)

A nu se crede că aceasta este o obligație din partea dvs. Nu! Titlul pe care l-ați dat vă va ajuta mai tîrziu la regăsirea acestui program, deoarece pe parcursul lucrării se vor aduna mai multe programe. EXEMPLUL 3 – PC(AMIC) reprezintă pentru program un comentariu sau o declarație. În limbajul BASIC orice comentariu se introduce prin instrucțiunea **REM**.

Formatul instrucțiunii este:

Format general
⟨număr linie⟩**REM**⟨mesaj⟩

unde ⟨mesaj⟩ reprezintă comentariul program.

Remarci

- Instrucțiunea **REM** poate apare oriunde în cadrul unui program.
- Se recomandă includerea între ghilimele a comentariului (vezi regula).

De notat că la execuția programului toate liniile care încep cu **REM** sînt ignorate. **REM**-urile prezintă importanță numai pentru cel care citește programul. Ele au un rol deosebit pentru documentarea programelor BASIC.

Vă propunem să listați în continuare, utilizînd instrucțiunea **REM**, funcțiunile programului:

```
20 REM "PROGRAMUL CITEŞTE RAZA (R)"
30 REM "A UNUI REZERVOR SFERIC,"
40 REM "CALCULEAZĂ ARIA (A),"
50 REM "VOLUMUL (V) ŞI TIPĂREŞTE:"
60 REM "RAZA, ARIA, VOLUMUL"
```

Instrucțiunea INPUT

Acum dorim să citim **(INPUT)** într-o variabilă numerică (R) valoarea razei rezervorului sferic. Cum primesc variabilele valori? Există mai multe modalități, și anume: 1) prin atribuire propriu-zisă (instrucțiunea **LET**); 2) prin introducere dinamică (instrucțiunea **INPUT**) și 3) prin introducere statică (instrucțiunile **READ** și **DATA**).

Remarci

- Între două atribuiri valoarea unei variabile rămîne constantă.
- În momentul lansării în execuție a unui program toate variabilele din program primesc valoarea zero.

Dacă în programele dvs. plasați instrucțiunea **INPUT**, calculatorul se va opri, va afișa un simbol (două puncte, semnul întrebării etc.) și va aștepta răspunsul dvs. Așadar, instrucțiunea **INPUT** reprezintă un excelent mod de introducere a datelor în calculator. Ea plasează valorile ce urmează a se introduce într-o variabilă. În instrucțiunea **INPUT** se precizează variabilele de intrare definite în tabela de variabile.

Formatul general al instrucțiunii este:

Format general
⟨număr linie⟩**INPUT**⟨variabilă⟩ {,⟨variabilă⟩}

unde, ⟨variabilă⟩ reprezintă orice variabilă admisă, indiferent de tip.

Valorile variabilei se introduc în mod conversaţional în timpul execuţiei programului.

Observaţie. Se recomandă utilizarea instrucţiunii **INPUT** ori de cîte ori datele necesare execuţiei unui program nu sînt foarte numeroase.

Într-o instrucţiune **INPUT** putem întîlni deopotrivă variabile numerice, alfanumerice etc. Întrebarea care se pune este "cum procedăm într-o astfel de situaţie?".

Pentru a elimina orice confuzie sau vreun eventual blocaj al programului (atunci cînd răspunsul la întrebare nu este corect calculatorul afişează un mesaj de eroare şi de nenumărate ori se blochează) se plasează înaintea instrucţiunii **INPUT** o instrucţiune **PRINT** cu texte explicative care să specifice numele variabilelor ce trebuie evaluate de la tastatură.

Consideraţi că avem suficiente elemente pentru programarea în BASIC a primului pas al algoritmului: "Citeşte R?" Dacă da, scrieţi următoarele două instrucţiuni:

```
70 PRINT "RAZA=";
80 INPUT R
```

Observaţie. Am pus " ; " (vezi linia 70) deoarece dorim ca la execuţia programului valoarea razei (R) să se introducă în continuarea mesajului: "RAZA=". În caz contrar, ar fi apărut pe rîndul următor.

După cum constataţi, de-abia acum am ajuns la momentul: "Conversaţia 2" (R=3 sau orice altă valoare doreaţi!). Pentru cei care au înţeles programul din conversaţia precedentă, nimic nu mai este complicat de aici înainte. Urmăriţi documentaţia de proiectare şi completaţi programul dvs. cu următoarele instrucţiuni:

```
90 A=4 * PI * R ↑ 2
100 V=4 * PI * R ↑ 3/3
110 PRINT "R="; R ; "A="; A; " V="; V
120 END
```

Introducerea programului (pasul 2). Reguli generale

Calculatoarele sînt virtual nefolositoare dacă nu ar exista modalităţi de introducere în memoria lor a datelor şi programelor noastre.

Calculatorul personal aMIC ia cunoştinţă de programul pe care i l-am stabilit atît prin folosirea unor medii magnetice (casete), cît şi prin intermediul unor medii "electronice" (console cu tastatură).

În vederea introducerii şi corectării interactive a programelor scrise în limbajul BASIC-aMIC se vor avea în vedere următoarele **reguli**:

- linia program BASIC poate fi tastată numai după apariţia cursorului " ■ ";
- pentru ca linia să fie prelucrată se va acţiona tasta **[RETURN]**;
- listarea instrucţiunilor unui program (toate sau numai o parte) se realizează cu comanda **LIST**;
- ştergerea ultimului caracter introdus se realizează prin apăsarea tastei **[DEL]**;

- ştergerea unei linii în curs de introducere se realizează acţionînd simultan [CTRL] şi [Y];
- ştergerea unei linii program se realizează prin scrierea numărului liniei, urmată de acţionarea tastei [RETURN];
- introducerea unei instrucţiuni cu un număr de linie deja existent în program are ca efect înlocuirea vechii linii program cu cea nouă;
- ştergerea din memorie a unui program pentru a face posibilă introducerea unui nou program se realizează cu comanda **SCR**;
- ştergerea ecranului în vederea scrierii sau desenării pe pagină nouă se realizează cu **INIT**.

Şi-acum, vă invităm la muncă! Tastaţi instrucţiunile programului respectînd întocmai regulile sintetizate mai sus. Succes!

Salvarea unui program BASIC pe casetă. Un program BASIC aflat în memorie poate fi salvat pe casetă magnetică utilizînd comanda (BASIC) **SAVE**. Procedura de salvare se realizează în cinci paşi, după cum urmează:

1) Se tastează **SAVE, [RETURN]**, patru caractere hexazecimale (numele programului) în această ordine.

2) Se poziţionează caseta în poziţia dorită.

3) Se apasă la casetofon clapele 1 (roşie) şi a 4-a (de la stînga la dreapta).

4) Se apasă tasta **[RETURN]**.

5) Apariţia mesajului **READY** dă de ştire că salvarea programului a reuşit.

Aplicaţie. Salvaţi programul pe o casetă audio sub numele PC3A.

Citirea de pe casetă a unui program BASIC. Procedura de citire a unui program BASIC înregistrat pe o casetă (cu comanda BASIC-**SAVE**) este următoarea:

1) Se tastează comanda **LOAD** urmată de **[RETURN]** şi de numele programului (patru caractere hexazecimale).

2) Se poziţionează caseta cît mai aproape de locul în care a fost înregistrat programul.

3) Se apasă clapele 4 şi 6 ale casetofonului.

4) Se eliberează clapa 6 şi se apasă pe **[RETURN]**.

5) Apariţia mesajului **READY** indică o citire de program reuşită.

6) În caz de eroare la citire, apare mesajul TAPE ERROR şi operaţiile trebuie reluate!

Aplicaţie. Citiţi de pe casetă programul PC3A salvat anterior. Dacă aţi reuşit, continuaţi. Dacă nu aţi reuşit, continuaţi!

Execuţia programului (pasul 3). Reguli generale

Procesul de execuţie a unui program trebuie înţeles astfel: 1) se transmite programul sursă, ca intrare, interpretorului BASIC; 2) interpretorul analizează fiecare instrucţiune, verifică existenţa erorilor, după care execută instrucţiunea BASIC solicitată. Pentru a executa instrucţiunea respectivă, interpretorul traduce instrucţiunea din codul ASCII într-un număr de instrucţiuni maşină executabile, pe care le execută în continuare.

Erorile pe care interpretorul le poate detecta se referă în general la erorile sintactice ale programului. Astfel de erori apar atunci cînd limbajul folosit în programul sursă nu este în conformitate cu regulile limbajului BASIC-aMIC. Fiecare eroare trebuie corectată astfel încît la o nouă execuţie interpretorul BASIC să nu mai semnaleze nici o eroare sintactică.

În vederea lansării în execuţie a unui program scris în limbajul BASIC-aMIC trebuie cunoscute şi respectate următoarele **reguli**:

- lansarea în execuţie a unui program începînd cu prima instrucţiune se realizează cu comanda **RUN**;
- lansarea în execuţie a unui program începînd cu orice instrucţiune se realizează cu comanda **RUN** [⟨număr linie⟩] sau cu instrucţiunea **GOTO** ⟨număr linie⟩;
- depistarea unei erori în timpul execuţiei unui program duce la apariţia mesajului:

  ERROR ⟨cod⟩ IN LINE ⟨număr linie⟩

  şi la abandonarea execuţiei programului;
  Lista mesajelor de erori se găseşte în vol. 2, pag. 2;
- întreruperea execuţiei unui program se realizează acţionînd tasta **[INT]**. Reluarea execuţiei se face prin apăsarea tastei [C].

Destul cu teoria! Să trecem la execuţia programului pe care sperăm că l-aţi încărcat de pe casetă. Tastaţi comanda **RUN**! A început dialogul. Pe ecran s-a afişat RAZA=, ca urmare a execuţiei instrucţiunii din linia 70.

```
RAZA = : ■
```

Nimic nou pînă acum. Între timp, după semnul egal au apărut două puncte. Le-aţi remarcat? Ele provin de la instrucţiunea **INPUT**. Ori de cîte ori se execută instrucţiunea **INPUT**, calculatorul afişează pe ecran " : " întrebînd: "ce valoare doriţi să atribuiţi variabilei R?"

Cum veţi răspunde dvs. la această ... provocare?

Tipăriţi o valoare numerică. Aţi tastat cumva 3, valoarea razei din conversaţia precedentă? Puteţi introduce, desigur, orice valoare doriţi!

După ce aţi introdus valoarea numerică a lui R (3, de exemplu), calculatorul aşteaptă pînă apăsaţi pe **[RETURN]**.

Calculatorul aMIC, ca de altfel toate calculatoarele, este foarte, foarte răbdător, el vă va aştepta oricît, staţionînd în linia 80 pînă cînd veţi tipări o valoare numerică şi veţi apăsa pe tasta **[RETURN]**.

Deci, apăsaţi pe **[RETURN]**. Calculatorul reia execuţia programului de la linia 90 începînd cu calculul ariei şi volumului rezervorului sferic şi continuînd apoi cu tipărirea valorilor calculate, specificate în linia 110.

```
RUN
R=3     A=113.097     V=113.097
```

În acest moment putem spune că programul este în întregime executat.

Acum puteți introduce o altă comandă **RUN,** răspunzînd la instrucțiunea **INPUT** cu orice valoare doriți. În acest fel vă puteți „juca" oricît doriți iar execuția programului nu va fi reluată pînă ce nu veți apăsa pe tasta **[RETURN].**

Observație. Comparînd acest program cu cel din conversația precedentă puteți constata avantajul net al acestuia. Prin utilizarea instrucțiunii dinamice (**INPUT**) de introducere a datelor (în acest program) nu mai este nevoie să specificați valorile numerice în cadrul programului (cu instrucțiunea **LET**). Valorile numerice sînt introduse de la claviatură, în timpul procesului de execuție a programului.

Bineînțeles, puteți experimenta cu acest program atîta timp cît el rezidă în memoria calculatorului. Introduceți doar comanda **RUN** ori de cîte ori doriți să reluați programul, răspunzînd apoi celor două puncte generate de instrucțiunea **INPUT,** cu valori numerice.

Aplicație. Scrieți un program BASIC-aMIC care să afișeze:

```
RUN
RAZA CERCULUI = 8
LUNGIMEA CERCULUI =
```

```
10 PRINT "RAZA CERCULUI = ";
20 INPUT R
30 L = 2 * PI * R
40 PRINT "LUNGIMEA CERCU-
   LUI= "; L
50 END
```

TESTE

a) Restrîngeți următoarele patru programe BASIC într-unul singur:

a1)
```
10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X+Y="; X+Y
60 END
```

a2)
```
10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X-Y="; X-Y
60 END
```

a3)
```
10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X+Y="; X+Y
60 END
```

a4)
```
10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X/Y="; X/Y
60 END
```

R.
```
10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X+Y="; X+Y
60 PRINT "X-Y="; X-Y
70 PRINT "X*Y="; X*Y
80 PRINT "X/Y="; X/Y
90 END
```

b) Scrieți un program BASIC care să calculeze și să afișeze valoarea expresiei A*B+C pentru anumite valori de

intrare ale lui A, B și C. Prezentați două variante.

b1)
```
10 INPUT A, B, C
20 D=A * B+C
30 PRINT "A * B+C="; D
40 END
```

b2)
```
10 INPUT A, B, C
20 PRINT "A*B+C="; A*B+C
30 END
```

c) Fiecare din următoarele instrucțiuni BASIC conține cîte o eroare. Identificați eroarea și scrieți corect fiecare instrucțiune în parte.

```
10 INPUT X; Y
20 "X+Y"; X+Y
30 LET X+2=Y
40 INPUT X, Y, Z,
50 INPUT, X, Y
60 PRINT "X+2=; X+2
70 IMPUT X, A
```

d) Un corp cu masa m=1,00 kg este aruncat orizontal cu viteza inițială v_0=20 m/s. Să se calculeze energia cinetică a corpului după un interval de timp Δt=4 s.

Indicație. Se va utiliza formula de calcul $E_c=\frac{1}{2}m(v_0^2+g^2\Delta t^2)$. Valorile variabilelor de intrare se vor introduce dinamic.

```
RUN
M, V0, T0 : 1, 20, 4
ENERGIA CINETICĂ = 969.8888 J
Ready
```

■

e) Precizați ce greșeală (de logică) s-a strecurat la introducerea următorului program BASIC care calculează aria și lungimea unui cerc de rază R.

```
10 REM PROGRAMUL CALCULEAZĂ
   ARIA (A) ȘI
20 REM LUNGIMEA (L) UNUI CERC
30 PRINT "RAZA=";
40 INPUT R
50 LET L=2 * PI * R
60 A=PI * R * 2
70 PRINT "R="; R, "L="; L,
   "A="; A
80 END
```

R. 60 A = PI * R ↑ 2

Variantă de utilizare a instrucțiunii INPUT

Cu o instrucțiune **INPUT** se pot atribui una, două sau mai multe valori variabilelor din cadrul programului.

```
10 PRINT "VALORILE LUI A SI B";
20 INPUT A, B
30 PRINT "A+B="; A + B
40 END
RUN
VALORILE LUI A SI B :5 ,9
A+B=14
```


Fig. 3.3.

În fig. 3.3 se prezintă un exemplu de utilizare a instrucțiunii **INPUT** pentru citirea dinamică a valorilor variabilelor A și B.

De notat că 5 este valoarea alocată primei variabile (A), iar 9 este valoarea alocată celei de-a doua variabile (B).

□ Codificarea în limbajul BASIC-PRAE

vol. 2, pag. 201

Scrierea programului. Reguli generale

Regulile generale de scriere (sintaxă) în limbajul BASIC-PRAE sînt următoarele:

- pentru scrierea unui program BASIC–PRAE se folosesc numai litere mari;
- fiecare linie-program începe cu un număr de linie;
- numărul de linie este un număr întreg format din 1 pînă la 5 cifre, putînd avea valoarea cuprinsă între 1 şi 65529;
- o linie program poate conţine una (linie simplă) sau mai multe (linie multi-instrucţiune) instrucţiuni separate între ele prin două puncte (" : ");
- numele variabilelor numerice simple sînt formate dintr-o literă urmată de oricîte caractere alfanumerice **(numai primele două caractere sînt luate în considerare)**;
- numele variabilelor indexate – idem;
- numărul de indici ai unei variabile indexate este de la 1 la 255;
- operatorii aritmetici admişi sînt: + (adunare); – (scădere); * (înmulţire); / (împărţire); ↑ (ridicare la putere);
- operatorii de relaţie admişi sînt: =; <; >; <=; >=; <>;
- operatorii logici admişi sînt: **NOT, AND, OR**;
- funcţiile matematice admise sînt: **PI, SIN, COS, TAN, ATN, LOG, EXP, SQR, ABS, INT, SGN**;
- funcţiile grafice admise sînt: **PLOT, PLOTC, DRAW, DRAWC, CIRCLE, CIRCLEC**;
- funcţiile pentru lucrul cu şirurile de caractere admise sînt: **LEFT$, RIGHT$, MID, VAL, STR, LEN, CHR$, ASC**;
- instrucţiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN ... ELSE, ON ... GOTO**;
- instrucţiunea de ciclare permisă este **FOR ... NEXT.**

Să aplicăm aceste reguli la scrierea programului. Imaginea acestuia (vezi vol. 2, pag. 201) este identică cu cea a programului precedent. În cele ce urmează vom preciza cîteva particularităţi privind instrucţiunile **REM** şi **INPUT**.

Remarci

- Pe o linie multiinstrucţiune, instrucţiunea **REM** trebuie să fie ultima instrucţiune a liniei (Exemplu: 10 A=2: **REM** INIŢIALIZARE). În caz contrar, textul instrucţiunii următoare se consideră ca făcînd parte din textul comentariului.
- În locul lui **REM** (trei litere) se poate utiliza apostroful, care are aceeaşi semnificaţie cu **REM.**
- Utilizarea apostrofului pentru comentarii oferă unele facilităţi în plus: poate ocupa prima poziţie într-o linie multiinstrucţiune şi poate termina o linie simplă sau multiplă (multiinstrucţiune) fără a fi precedat de caracterul de separare instrucţiuni " : " (Exemplu: 10 A=4*PI*R↑2 'ARIA SFEREI).
- În BASIC-PRAE există posibilitatea cuplării instrucţiunilor **PRINT** şi **INPUT.** Începînd cu conversaţia următoare vom folosi şi noi această facilitate importantă a interpretorului.

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii interactive a programelor BASIC-PRAE sînt:

- linia program poate fi tastată numai după apariţia cursorului "←";
- pentru ca linia să fie prelucrată se va acţiona tasta **[CR]**;
- listarea instrucţiunilor unui program (toate sau numai o parte) se realizează cu comanda **LIST**;
- ştergerea ultimului caracter introdus se realizează prin acţionarea tastei **[DEL]**;
- ştergerea unei linii program se realizează prin scrierea numărului liniei, urmată de acţionarea tastei **[CR]**;
- o metodă foarte eficace de a corecta greşelile din liniile unui program este folosirea funcţiei **EDIT**;
- ştergerea din memorie a unui program pentru a face posibilă introducerea unui nou program se realizează cu comanda **NEW**;
- ştergerea ecranului se realizează cu instrucţiunea **CLS**;
- generarea automată a numerelor de linie se face cu comanda **AUTO** al cărei format general este:

Format general
AUTO[⟨nr. linie⟩[,⟨pas⟩]

Observaţie. Dacă ⟨nr. linie⟩ şi ⟨pas⟩ lipsesc, se generează numere de linie începînd cu 10 şi pasul 10. Revenirea în modul de numerotare manuală se face apăsînd de două ori pe **[CR]**;

- Renumeratoarea liniilor se face cu comanda **RENUMBER** al cărei format general este:

Format general
RENUMBER[⟨nr. linie⟩, ⟨pas⟩, ⟨nr. linie minim⟩]

unde: ⟨nr. linie⟩ reprezintă numărul de linie de la care se începe renumerotarea; ⟨pas⟩ este mărimea pasului dintre numerele de linie atribuite la două linii consecutive, iar ⟨nr. linie minim⟩ reprezintă numărul de linie minim ce se renumerotează.

Salvarea unui program pe casetă magnetică

Pentru păstrarea programelor BASIC-PRAE avem nevoie de un casetofon obişnuit (DECK EM-1001, Dana etc.) şi de o casetă tot din cele obişnuite (se pot folosi şi magnetofoanele). Procedura de salvare a unui program este următoarea:

1) Se tastează **ASAVE** ⟨nume⟩, unde ⟨nume⟩ reprezintă numele programului ales de utilizator, format din maximum 16 caractere.

2) Se poziţionează caseta în punctul dorit. Înainte de a acţiona [**CR**] se pot tasta cîteva blancuri pentru a verifica nivelul semnalului de înregistrare la casetofon (trebuie să fie în jur de zero decibeli). Dacă nivelul este reglat corect se poate porni casetofonul şi apoi se poate tasta [**CR**].

3) Se apasă clapele 1 (roşie) şi 4 (de la stînga la dreapta).

4) Se tastează **[CR]**.

5) În timpul înregistrării se aude un sunet specific. Cînd calculatorul PRAE a terminat transmiterea programului, pe ecran apare mesajul "TRY

TO READ" ca o invitaţie de a încerca să citim ceea ce am înregistrat pe casetă. Se apasă clapele [5], [2], [5], [4], **[CR]** în această ordine.

6) Corectitudinea salvării este indicată de apariţia mesajului **READY**. În cazul apariţiei unei erori la înregistrare, se afişează mesajul "TAPE ERROR". În această ultimă situaţie, ce reprezintă un caz nefericit, trebuie să încercăm salvarea programului cu un alt nivel de înregistrare al casetofonului, sau, eventual, pe o altă porţiune a casetei magnetice.

Aplicaţie. Salvaţi programul pe o casetă audio, sub numele: PC3PRAE.

Citirea de pe casetă a unui program BASIC

Citirea unui program BASIC înregistrat pe o casetă se face cu comanda **ALOAD** al cărei format general este:

Format general
ALOAD "⟨nume⟩"

După acţionarea tastei **[CR]** calculatorul PRAE este pregătit să găzduiască programul. Pot să apară două situaţii: a) pe casetă se găseşte programul chemat cu comanda **ALOAD;** b) pe casetă se găseşte un program diferit de cel specificat în comanda **ALOAD.** În cel de-al doilea caz, PRAE afişează numele acestuia. Cînd calculatorul găseşte programul solicitat (cazul a), el procedează imediat la încărcarea programului (blocurilor). Aşteptaţi cîteva secunde pînă la apariţia mesajului **READY** (timpul necesar conversiei). Dacă în timpul încărcării programului s-a detectat undeva o eroare, veţi citi mesajul "TAPE ERROR" şi după cîteva secunde se trece în **READY.** În cazul cînd nu v-a reuşit prima tentativă de încărcare a programului, "număraţi pînă la zece" (nimic mai mult!) şi reluaţi operaţiunea de la comanda **ALOAD.** Cauzele nereuşitei se pot datora fie stării necorespunzătoare a capului de înregistrare, fie vitezei casetei magnetice sau chiar unor deteriorări ale acesteia.

Şi-acum, pe scurt, procedura de citire a unui program BASIC înregistrat pe o casetă:

1) Se tastează comanda **ALOAD.**

2) Se poziţionează caseta cît mai aproape posibil de locul în care a fost înregistrat programul.

3) Se tastează clapele [4], [6] ale casetofonului în această ordine.

4) Pe ecran se listează numele tuturor programelor existente pînă la programul căutat.

5) La o citire corectă a programului se va afişa mesajul **READY.** În caz de eroare, vă va întîmpina mesajul TAPE ERROR, iar operaţiile trebuie reluate.

Metodă specială de salvare a unui program BASIC-PRAE

Pentru salvarea programelor BASIC-PRAE deosebit de lungi, se recomandă [36] utilizarea comenzii **SAVE** al cărei format general este:

Format general
SAVE ⟨nume program⟩&4100, **PEEK**(&415E)+**PEEK**(&415F)*256+1

Cu această comandă se salvează o zonă de memorie care începe cu primele variabile de sistem BASIC și se termină cu adresa de sfîrșit a programului BASIC. Metoda se aplică pentru calculatoarele PRAE cu aceeași capacitate de memorie (variabilele de sistem BASIC diferă de la un PRAE cu 16 K de cele de la un PRAE cu 48 K).

Cu comanda **SAVE** puteți salva un program BASIC împreună cu variabilele sale simple. Formatul general este:

Format general

SAVE ⟨nume program⟩&4100, **PEEK**(&4160)+**PEEK**(&4161)*256+1

Comenzile pentru salvarea și încărcarea imaginilor de pe ecran au următoarele formate generale:

Format general

SAVE ⟨nume&6000⟩, &8000

Format general

SAVE ⟨nume&E000⟩, &FFFF

Format general

LOAD ⟨nume&6000⟩

Format general

LOAD ⟨nume&E000⟩

Remarci

- Comanda **ASAVE** convertește programul curent aflat în memorie în cod **ASCII** și îl salvează pe suportul extern (casetă magnetică).
- Comanda **ALOAD** permite încărcarea în memoria calculatorului a programelor scrise în cod ASCII. Fiecare linie începe cu un număr de linie și se termină cu [**CR**] (retur de car). Sfîrșitul se detectează fie prin codul CTRL/Z din text, fie prin marcajul EOF.
- Comanda **SAVE** produce salvarea blocului de memorie curent din memoria internă pe suport extern. Formatul general este:

Format general

SAVE ⟨nume fișier⟩ ⟨adresa de la care se salvează⟩ ⟨adresa pînă la care se salvează⟩

- Comanda **LOAD** servește la încărcarea unui program la ⟨adresă început⟩ de pe un suport extern indicat.

Format general

LOAD ⟨nume fișier⟩ ⟨adresă început⟩

- Un program salvat cu **SAVE** nu poate fi încărcat cu comanda **ALOAD.**
- Un program salvat cu **ASAVE** poate fi încărcat cu **ALOAD.**
- Comanda **AMERGE** serveşte pentru încărcarea şi interclasarea unui program de pe casetă cu programul aflat în memorie. În timpul căutării programului ⟨nume fişier⟩ se listează numele tuturor programelor întîlnite pe suport.

Format general
AMERGE ⟨nume fişier⟩

Aplicaţie. Citiţi de pe casetă programul PC3PRAE, respectînd regulile listate anterior.

Execuţia programului. Reguli generale

Pentru lansarea în execuţie a unui program BASIC-PRAE, trebuie cunoscute şi respectate următoarele **reguli** generale:

- Comanda **RUN** realizează lansarea în execuţie a programului curent existent în memoria internă a calculatorului, de la instrucţiunea cu cel mai mic număr de linie;
- comanda **RUN** realizează lansarea în execuţie a programului curent de la instrucţiunea care are numărul de linie egal cu ⟨nr. linie⟩;

Format general
RUN ⟨nr. linie⟩

- comanda **RUN** poate fi lansată şi dintr-un program dacă este precedată de un număr de linie.

Format general
⟨nr. linie⟩ **RUN**

- întreruperea execuţiei unui program se realizează acţionînd tasta **[CTRL]**. Reluarea execuţiei se face prin apăsarea tastei **[C]**;
- lista mesajelor de eroare BASIC-PRAE se găseşte în vol. 2, pag. 3.
- comanda **CONTINUE** permite reluarea execuţiei programului din punctul de întrerupere realizat prin instrucţiunea **STOP**;
- comanda **CONTINUE** nu poate înlocui comanda **RUN** şi nu are sens decît în punctele de întrerupere ale unui program lansat prin **RUN.**

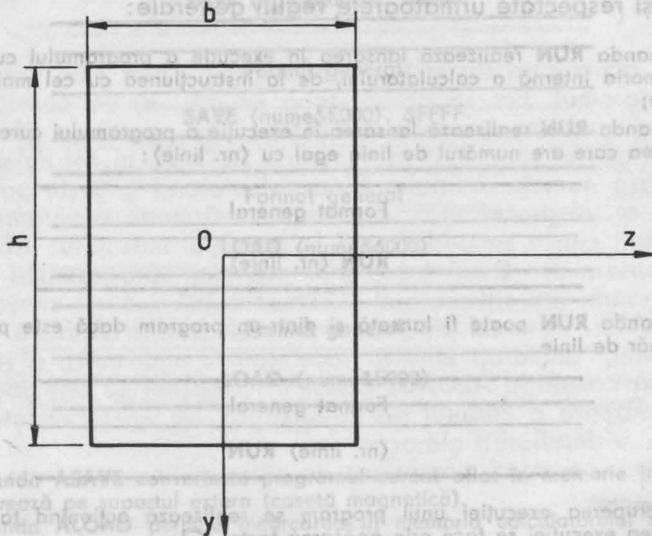
Aţi încărcat programul de pe casetă în memoria calculatorului? Dacă nu, vă rugăm s-o faceţi acum. Tastaţi apoi **RUN** şi urmăriţi imaginea de pe ecran. După ce calculatorul a afişat "RAZA=", remarcaţi cum, imediat, în continuarea mesajului, a apărut un semn de întrebare. Din acest moment calculatorul aşteaptă ... valoarea razei rezervorului, pe care vă rugăm s-o introduceţi tastînd 3, **[CR]** în această ordine. Dacă vă permiteţi fantezii şi introduceţi ceva diferit de un număr (nu neapărat 3!), veţi primi mesajul "INVALID INPUT" şi din nou îşi va face apariţia prietenul nostru curios — semnul întrebării.

RAZA =?

Remarcaţi pe rîndul următor valorile ariei şi volumului rezervorului sferic. Între timp a apărut şi **READY** care parcă vrea să spună "pe cînd o nouă conversaţie?"

Aplicaţie. Să se calculeze momentele de inerţie axiale (Iz, Iy) pentru suprafaţa dreptunghiulară din figura de mai jos, în raport cu axele Oz şi Oy.

Indicaţie. Se vor utiliza formulele de calcul: $Iz = \frac{bh^3}{12}$; $Iy = \frac{b^3h}{12}$.



Specificaţiile de programare şi documentaţia de proiectare sînt ilustrate în modulele de analiză şi proiectare structurată.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieşire
B: lăţimea		IZ: momentul de inerţie axial în raport cu axa OZ
H: lungimea		IY: momentul de inerţie axial în raport cu axa OY

a)

Fig. 3.4. Modulele de analiză şi proiectare structurată: *a)* tabela de variabile;

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal lungimea și lățimea secțiunii, calculează momentele de inerție axiale și afișează rezultatul.

Intrări

Lungime, lățime

Ieșiri

Momentele de inerție axiale IZ, IY

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează momentul de inerție în raport cu axa OZ
3. Calculează momentul de inerție în raport cu axa OY
4. Afișează rezultatul
5. Stop.

b)

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2,3
LIST	4
SFÎRȘIT	5

c)

DATELE DE INTRARE

B	H
20*	60*

* în cm.

d)

APLICATIE

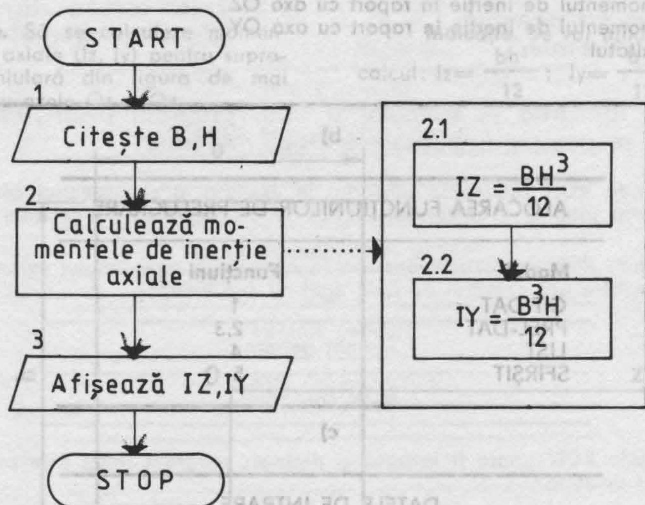
CIT-DAT | PREL-DAT | LIST | SFIRSIT

e)

Fig. 3.4. a–g. Modulele de analiză și proiectare structurată: *b*) specificații de programare; c) alocarea funcțiunilor de prelucrare; *d*) datele de intrare; e) diagrama de structură;

	PSEUDOCODUL
APLICAȚIE	**SEQ**
	INPUT B, H
PREL-DAT	**SEQ**
	IZ=(B*H ↑ 3)/12
	IY=(B ↑ 3*H)/12
PREL-DAT	**END**
	PRINT "IZ="; IZ, "IY="; IY
APLICAȚIE	**END**

f)



g)

f) pseudocodul; *g)* schema logică.

```
10 REM PROGRAMUL CALCULEAZĂ MOMENTELE
20 REM DE INERȚIE AXIALE PEN-
30 REM TRU SUPRAFAȚA DREPT-
   UNGHIULARĂ
40 PRINT "B, H";
50 INPUT B, H
60 IZ=(B * H ↑ 3)/12
70 IY=(B ↑ 3 * H)/12
80 PRINT "IZ="; IZ, "IY="; IY
90 END
RUN
B, H? 20, 60
IZ=   360000   IY=   40000
```

TEST

Scrieți un program BASIC-PRAE care să afișeze:

```
RUN
GENERATOAREA CILINDRULUI = ? 8
RAZA CILINDRULUI = ? 2
ARIA LATERALĂ =
VOLUMUL=
```

```
10 PRINT "GENERATOAREA
   CILINDRULUI = ";
20 INPUT G
30 PRINT "RAZA CILINDRULUI" = ";
40 INPUT R
50 AL=2 * PI * R * G
60 AT=AL+2 * PI * R ↑ 2
70 V=PI * R ↑ 2 * G
80 PRINT "ARIA LATERALĂ="; AL
90 PRINT "ARIA TOTALĂ="; AT
100 PRINT "VOLUMUL="; V
110 END
```

☐ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 202

Scrierea programului. Reguli generale

Limbajul BASIC HC-85, TIM S, SPECTRUM dispune de următoarele **reguli** generale de scriere:

- fiecare linie program începe cu un număr de linie;
- numărul de linie este un număr întreg format din 1 pînă la 4 cifre, putînd avea valoarea cuprinsă între 1 și 9999;
- o linie program poate conține una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- numele variabilelor numerice simple sînt formate dintr-un număr nelimitat de caractere (litere, cifre, blanc) care încep cu o literă. Litera mare și litera mică omoloagă sînt interpretate la fel;
- variabilele de tip șir de caractere sînt formate dintr-o literă urmată de "$";
- variabilele utilizate în instrucțiunea **FOR** sînt formate dintr-o singură literă;
- numele variabilelor indexate sînt formate dintr-o literă;
- operatorii aritmetici admiși în limbajul BASIC HC-85, TIM S, SPECTRUM sînt: +; −; *; /; ↑;
- operatorii de relație admiși sînt: <; >; <=; >=; <>;
- operatorii logici admiși sînt: **NOT, AND, OR**;
- funcțiile aritmetice admise sînt: **SIN, COS, TAN, ATN, LN, EXP, SQR, ABS, INT, SGN, ASN, ACS**;
- funcțiile grafice admise sînt: **PLOT, DRAW, CIRCLE, POINT**;
- instrucțiunile admise pentru utilizarea culorilor sînt: **PAPER, INK, FLASH, INVERSE, OVER, BORDER, ATTR**;
- funcțiile care operează cu șiruri de caractere sînt: **LEN, STR$, VAL, RIGHT$, MID$, CHR$, INKEY$**;
- alte instrucțiuni și funcții disponibile sînt: **USR, BIN, RANDOMIZE, RND, DEF**;
- funcțiile de intrare admise sînt: **INKEY$, PEEK**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN**;
- instrucțiunea de ciclare permisă este **FOR ... NEXT**;
- instrucțiunile de intrare/ieșire permise sînt: **INPUT, INPUTLINE, PRINT, LPRINT, READ, DATA, RESTORE, IN, OUT, POKE**;
- instrucțiunea admisă pentru generarea sunetelor este **BEEP.**

Vom aplica aceste reguli la scrierea programului.

```
10 REM PROGRAMUL CITEŞTE RAZA
   (R) A UNUI REZERVOR ŞFERIC,
   CALCULEAZĂ ARIA (A), VOLUMUL (V)
   ŞI TIPĂREŞTE: RAZA, ARIA, VOLUMUL    V
20 PRINT "RAZA=";
30 INPUT R
40 LET A=4 * PI * R ↑ 2
50 LET V=4 * PI * R ↑ 3/3
60 PRINT "R="; R; "A="; A; "V=";
70 STOP
```

Remarcați modul de scriere a comentariului din linia 10 — pe patru linii (fizice) ecran. Pentru tipărirea mesajului "RAZA=" am folosit instrucțiunea **PRINT** cu separatorul punct și virgulă (vezi linia 20). Instrucțiunea **INPUT** (din linia 30) citește valoarea razei (în variabila R) în timpul execuției programului.

Observație. Instrucțiunile de atribuire din liniile 40 și 50 folosite pentru calculul ariei și volumului rezervorului cer folosirea obligatorie a cuvîntului **LET.**

Pentru afişarea rezultatului final s-a scris instrucţiunea **PRINT** din linia 60 care utilizează pentru tabulare separatorul " ; ". În sfîrşit, instrucţiunea **STOP** (vezi linia 70) opreşte execuţia programului.

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii şi corectării interactive a programelor BASIC HC-85, TIM S, SPECTRUM sînt următoarele:

- linia program poate fi tastată numai după apariţia literei de cursor;
- litera de cursor (care arată unde va fi inserat următorul caracter tastat) poate fi: **K, L, C, E, G.** Cele cinci litere cursor se identifică cu modurile de lucru ale calculatorului;
- literele de cursor **K, L, C, E** sau **G** pot fi obţinute prin acţionarea tastelor [**CAPS SHIFT**], [**SYMBOL SHIFT**];
- modul **K** se formează automat la începutul liniei după caracterul " : " sau după **THEN**;
- în modul **L** (litere mici) se afişează simbolul principal de pe tastă iar literele se editează cu minuscule. [**CAPS SHIFT**] cu o tastă literă editează litera mare în modul **L**.
- în modul **C** (litere mari), literele se afişează cu majuscule. [**CAPS LOCK**] realizează trecerea **L** la **C** şi **C** la **L**;
- cursorul **E** apare prin tastarea simultană a [**CAPS SHIFT**] şi [**SYMBOL SHIFT**];
- modul **E** (extindere) este utilizat pentru realizarea semnificaţiei scrise deasupra tastei cu albastru (TIM S) prin apăsarea tastei sau pentru realizarea semnificaţiei scrise dedesubtul tastei cu roşu (TIM S) prin tastarea simultană a tastei alese şi [**SYMBOL SHIFT**];
- în modul **G** (tastaţi 9 şi [**CAPS SHIFT**]) – semigrafic şi grafic tastarea cifrelor va edita caracterele semigrafice înscrise pe taste, iar tastarea literelor de la A la T va edita caractere grafice definite de utilizator.
- liniile se editează pe liniile inferioare ale ecranului alfanumeric;
- funcţiile de editare pentru un calculator cuplat cu un televizor alb/negru sînt realizate de tastele numerice apăsate simultan cu [**CAPS SHIFT**];
- ştergerea ultimului caracter introdus se realizează prin acţionarea tastei [**DELETE**];
- comenzile **TRUE VIDEO** şi **INV VIDEO** editează caracterul ce urmează a fi tastat în pozitiv şi negativ;
- comenzile >, < deplasează cursorul în sensul dorit fără a şterge textul;
- pentru ca linia să fie prelucrată, se va acţiona tasta [**ENTER**];
- linia de program (precedată de un număr de linie) va fi plasată în textul editat în partea superioară a ecranului;
- dacă linia este incorectă, în linia de editare va apare semnul întrebării;
- ultima linie editată are plasaţ între numărul de linie şi restul liniei simbolul "<" numit şi cursor de linie editată;
- pentru modificarea unei linii de program se deplasează cursorul linie la linia respectivă (cu ajutorul "∧", "V"), după care se tastează **EDIT.** Linia va fi adusă în spaţiul de editare unde se modifică tastînd <, > şi [**DELETE**]. În final, se tastează [**ENTER**];
- comanda **NEW** iniţializează sistemul BASIC ştergînd programul sau variabilele curente;
- comenzile **LIST** şi **LIST** ⟨nr. linie⟩ afişează pe ecran textul programului BASIC începînd cu cel mai mic număr de linie, respectiv ⟨nr. linie⟩. La umplerea ecranului (instrucţiunea listează 22 linii începînd din partea de sus a ecranului şi ⟨nr. linie⟩ linie curentă), în partea de jos a acestuia se afişează "scroll?" (dacă se tastează "N" se opreşte afişarea);
- într-o instrucţiune de atribuire, cuvîntul **LET** este obligatoriu;
- o variabilă simplă şi o variabilă tablou pot avea acelaşi nume;
- listarea la imprimantă se realizează cu instrucţiunile **LLIST** şi **LPRINT** care au aceleaşi funcţii ca **LIST** şi **PRINT.** Nu va mai apare mesajul "scroll".

Salvarea unui program BASIC pe casetă

Pentru păstrarea programelor BASIC HC-85, TIM S, SPECTRUM avem nevoie de un casetofon audio uzual și de casete audio obișnuite (casetele audio de 60 minute asigură un spațiu de înregistrare de circa·900 K). Procedura de salvare [45] a unui program este următoarea:

1) se tastează **SAVE** care are formatul:

Format general
SAVE "⟨nume⟩"

unde, ⟨nume⟩ reprezintă numele programului ales de utilizator, format din maximum 10 caractere;

2) se poziționează banda în punctul unde se dorește înregistrarea, se pornește casetofonul și se apasă oricare tastă (calculatorul răspunde cu mesajul: "START TAPE THEN PRESS ANY KEY"). În timpul înregistrării se aude un sunet specific în difuzorul casetofonului. Corectitudinea salvării este indicată de apariția mesajului O.K.

REMARCI

- Pe o casetă audio obișnuită se pot salva patru tipuri de informații: 1) programul BASIC împreună cu variabilele sale; 2) tablouri numerice; 3) tablouri de caractere; 4) șiruri de biți (program în limbaj de asamblare sau imagini binare). Formatele generale ale comenzii **SAVE** (cu semnificația respectivă) sînt:

Format general	Semnificație
SAVE"⟨nume⟩"	Salvează programul BASIC și variabilele lui în fișierul ⟨nume⟩.
SAVE"⟨nume⟩"**LINE**⟨n⟩	Idem, cu deosebirea că, la încărcare, programul intră auotmat în execuție de la linia ⟨n⟩.
SAVE"⟨nume⟩"**CODE**"⟨start⟩","⟨lungime⟩"	Memorează șirul de biți de la adresa ⟨start⟩ pe ⟨lungime⟩ octeți în fișierul ⟨nume⟩.
SAVE"⟨nume⟩"**DATA**⟨litera()⟩	Memorează tablouri numerice ⟨litera⟩ în fișierul ⟨nume⟩.
SAVE"⟨nume⟩"**DATA**⟨litera$()⟩	Memorează "șir" ⟨litera $⟩ în fișierul ⟨nume⟩
SAVE"⟨nume⟩"**SCREEN$**	Salvează imaginea binară de pe ecran în fișierul ⟨nume⟩.

- Pentru verificarea informațiilor de pe casetă cu informațiile aflate deja în memorie, se utilizează comanda **VERIFY.**
- În timpul căutării programului specificat, calculatorul afișează numele tuturor programelor întîlnite.
- **VERIFY** are același format și aceeași semnificație ca **LOAD** numai că realizează doar o comparare cu programul din memorie. Dacă în urma comparării informațiile sînt identice se afișează mesajul "O.K.", iar în caz de neconcordanță se listează mesajul "R TAPE LOADING ERROR".

Aplicație. Salvați programul pe o casetă audio sub numele: EX3PCHCTS. Verificați informațiile de pe casetă cu informațiile aflate deja în memorie.

Citirea de pe casetă a unui program BASIC. Citirea unui program BASIC HC-85, TIM S, SPECTRUM înregistrat pe o casetă se face cu comanda **LOAD** al cărei format general este:

Format general
LOAD"⟨nume⟩"

De reţinut că această comandă şterge vechiul program aflat în calculator împreună cu variabilele sale înainte de a încărca unul nou.

Formatele generale ale comenzii **LOAD** (cu semnificaţia respectivă) [45] sînt:

Format general	Semnificaţie
LOAD ""	Încarcă de pe casetă primul fişier întîlnit.
LOAD "⟨nume⟩" **DATA** ⟨litera()⟩	Şterge orice tablou cu numele ⟨litera()⟩ din programul rezident în memoria calculatorului şi încarcă în loc tot ce se găseşte în fişierul ⟨nume⟩ de pe casetă.
LOAD "⟨nume⟩" **DATA** ⟨litera$()⟩	Şterge orice tablou cu numele ⟨litera$()⟩ din programul rezident în memoria calculatorului şi încarcă în loc tot ce se găseşte în fişierul ⟨nume⟩ de pe casetă.
LOAD "⟨nume⟩"**CODE** **LOAD** "⟨nume⟩"**CODE**"⟨start⟩" **LOAD** "⟨nume⟩"**CODE**"⟨start⟩""⟨lungime⟩"	Încarcă un şir de biţi din fişierul ⟨nume⟩ în memoria calculatorului începînd de la adresa ⟨start⟩ (pe un număr de octeţi ⟨lungime⟩) scriind peste ceea ce se găseşte în memorie.
LOAD "⟨nume⟩" **SCREEN$**	Încarcă fişierul ⟨nume⟩ în zona de memorie alocată imaginii ecran. *Observaţie.* Comanda este sinonimă cu **LOAD** "⟨nume⟩" **CODE** 16384, 6912.

Remarci

- La memorarea imaginii video nu poate fi folosită comanda **VERIFY.**
- Comanda **MERGE** încarcă un program înregistrat pe casetă în memoria calcu-

Format general
MERGE "⟨nume⟩"

latorului (ca şi **LOAD**), dar, spre deosebire de comanda **LOAD,** ea şterge din liniile şi variabilele vechiului program aflat în memorie doar pe cele care se suprapun ca număr sau ca nume cu cele din programul aflat în fişierul "⟨nume⟩". De reţinut că fişierul "⟨nume⟩" conţine numai programe scrise în limbajul BASIC. (n trebuie să fie cuprins între 0 şi 65535). Pentru **PAUSE** 0 calculatorul opreşte definitiv execuţia programului, iar 65535 corespunde la aproximativ 1320 secunde. Pentru continuarea la oprirea cu **PAUSE 0** se acţionează orice tastă.

Aplicaţie. Citiţi de pe casetă programul EX3PCHCTS salvat anterior.

Execuţia programului. Reguli generale

Dacă programul nu se află în memoria calculatorului, vă invităm şi de această dată să-l încărcaţi de pe casetă (vezi aplicaţia de la pag. 91 pe care trebuia s-o realizaţi singuri!) şi apoi să-l executaţi. Să urmărim împreună ce se petrece pe ecranul monitorului:

```
RAZA=?3
R=3 A=113.09734 V=113.09734

9 STOP statement,70:1
```

După ce linia 10 a fost ignorată de calculator, ca urmare a execuţiei instrucţiunii din linia 20 s-a tipărit mesajul: "RAZA=" şi imediat după aceea, în continuarea mesajului, şi-a făcut apariţia o mai veche cunoştinţă de-a noastră — semnul întrebării. Instrucţiunea **INPUT** din linia 30 aşteaptă să introduceţi o valoare pentru variabila R. Tastaţi 3 şi acţionaţi **[ENTER]**. Calculatorul afişează rezultatul şi se opreşte din execuţie cu eroare 9, datorită instrucţiunii **STOP** (linia 70). Programul se poate continua (în cazul nostru nu este nevoie deoarece **STOP** este ultima instrucţiune din program) tastînd de două ori **CONTINUE.**

Pentru lansarea în execuţie a unui program BASIC HC-85, TIM S, SPECTRUM trebuie cunoscute şi respectate următoarele **reguli** generale:

- comanda **RUN** realizează lansarea în execuţie a programului curent existent în memoria calculatorului, de la instrucţiunea cu cel mai mic număr de linie;
- comanda **RUN** ⟨nr. linie⟩ lansează în execuţie programul curent existent în memoria calculatorului de la instrucţiunea care are numărul de linie egal cu ⟨nr. linie⟩;
- comanda **GOTO** al cărei format general este:

Format general

GOTO ⟨nr. linie⟩

provoacă execuţia programului de la linia ⟨nr. linie⟩ fără a şterge nimic din memorie;

- comanda **RUN** poate fi lansată şi dintr-un program dacă este precedată de un număr de linie;
- întreruperea execuţiei unui program se realizează acţionînd tasta **[BREAK] (CAPS SHIFT şi SPACE).** Calculatorul răspunde cu mesajul: "L BREAK INTO PROGRAM". Apăsarea tastei **[CONTINUE]** reia execuţia programului cu următoarea instrucţiune BASIC;
- lista mesajelor de eroare BASIC se găseşte în vol. 2, pag. 5.
- instrucţiunea **STOP** are acelaşi efect ca şi tasta **[BREAK]**. Execuţia se poate relua tastînd **CONT** (CONTINUE);
- după introducerea datelor prin instrucţiunea **INPUT** se tastează **[ENTER]**;
- **PAUSE** al cărei format general este:

Format general
PAUSE n

opreşte execuţia programului şi aşteaptă un număr de n/50 secunde.

TESTE

a) Scrieţi un program BASIC HC-85, TIM S, SPECTRUM care să afişeze:

RUN
RAZA, ÎNĂLŢIMEA CONULUI = ? 3, 8
VOLUMUL CONULUI =

b) Pentru calculul expresiei:

$$E=\frac{n^2(n+1)^2}{4}-\frac{n(n+1)(2n+1)}{6}-\frac{n(n+1)}{2}$$

s-a scris un program BASIC în trei variante (b_1, b_2, b_3). Precizaţi care dintre versiuni este corectă.

b1)
```
10 PRINT "N=";
20 INPUT N
30 LET U=N*(N+1)/2
40 LET E=U*U-U*(2*N+1)/3-U
50 PRINT "E="; E
60 END
```

b2)
```
10 PRINT "N=":
20 INPUT N
30 LET E=(N*(N+1)/2)*
(N*(N+1)/2 - (2*N+1)/3-1)
40 PRINT "E="; E
50 END
```

b3)
```
10 PRINT "N=";
20 INPUT N
30 LET U=N*(N+1)/2
40 LET E=U↑2-U*(2*N+1)/3-U
50 PRINT "E=";, E
60 END
```

R. Toate trei.

c) Precizaţi rezultatul execuţiei următorului program BASIC HC-85, TIM S, SPECTRUM, dacă lui A şi B li se atribuie dinamic valorile 820, respectiv 105.

```
5 REM ** INPUT **
10 INPUT A
20 PRINT , A
30 INPUT B
40 PRINT , B
50 LET C=A+B
60 PRINT , "___"
70 PRINT , C
80 STOP
```

R. **RUN**
```
? 820
                820
? 105
                105
                ---
                925
9 STOP statement , 80 : 1
```

□ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 202

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-AMSTRAD (6128) trebuie respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- o variabilă este identificată printr-un șir de caractere care începe cu o literă, urmată de un număr oarecare de litere sau cifre;
- variabilele admise sînt de următoarele tipuri: numerice reale, numerice întregi, șiruri de caractere;
- tipul unei variabile se poate stabili și prin caracterele speciale: " ! " (pentru variabilele numerice reale); " % " (pentru variabilele numerice întregi); " $ " (pentru șiruri de caractere) ce se atașează numelui variabilei;
- o variabilă poate fi declarată și implicit cu ajutorul instrucțiunilor admise: **DEFINT, DEFREAL, DEFSTER**;
- dacă precizarea tipurilor implicite nu este făcută de utilizator, toate variabilele se consideră numerice reale;
- numele unei variabile indexate este orice nume de variabilă acceptat de către BASIC-AMSTRAD (6128);
- valoarea minimă a indicelui unui tablou este zero;
- operatorii aritmetici admiși sînt: " + " (pentru adunare); " − " (pentru scădere); "*" (pentru înmulțire); "/" (pentru împărțire); " ↑ " (pentru ridicare la putere);
- operatorii de relație admiși sînt: "="; "<"; ">"; "<="; ">="; "<>";
- operatorii logici admiși sînt: **AND, OR, XOR; NOT**;
- funcțiile matematice admise sînt: **SIN, COS, TAN, ATN, PI, SQR, EXP, LOG, LOG10, INT, ABS, SGN, FIX, CREAL, CINT, ROUND, RND, RANDOMIZE;**
- funcțiile admise pentru șirurile de caractere sînt: **LEN, POS, VPOS, INSTR, MID$, LEFT$, RIGHT$, STRING$, SPACE$, LOWER$, UPPER$, COPYCHR$**;
- funcțiile de conversie admise sînt: **CHR$, VAL, STR$, ASC, HEX$**;
- alte funcții (auxiliare) disponibile sînt: **TAB, ERR, ERROR, DERR, SPC, ZONE, EOF, TIME, MAX, MIN, MOD, MODE**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN ... ELSE, IF ... GOTO, ON ERROR GOTO, ON GOTO**;
- instrucțiunile de ciclare permise sînt: **FOR ... NEXT, WHILE ... WEND**;
- instrucțiunile de intrare/ieșire permise sînt: **READ, DATA, RESTORE, INPUT, LINE INPUT, PRINT, PRINT SPC, PRINT TAB, PRINT USING, POKE, PEEK**;
- instrucțiunile admise pentru operațiile de intrare/ieșire pe disc sînt: **OPENIN, OPENOUT, CLOSEIN, CLOSEOUT, INPUT, LINE INPUT, PRINT, PRINT USING, WRITE, SAVE, LOAD, RUN**;
- instrucțiunile grafice admise sînt: **LOCATE, WINDOW, MOVE, MOVER, ORIGIN, XPOS, YPOS, GRAPHICS PEN, GRAPHICS PAPER, INK, MASK, TAG, TAGOFF, DRAW, DRAWR, CLG, CURSOR, PLOTR**;
- instrucțiunile admise pentru generarea sunetelor sînt: **SOUND, ENT, ENV, ON SQ GOSUB.**

Vom aplica aceste reguli la scrierea programului.

Variabile cu același nume dar de tipuri diferite

După cum constatați, prima linie (10) am „sacrificat-o" pentru documentarea programului.

Observație. In BASIC-AMSTRAD, **REM** poate fi înlocuit printr-un apostrof.

Pentru tipărirea pe ecran a mesajului "RAZA=" am scris instrucțiunea 20. Instrucțiunea **INPUT** din linia 30 citește (dinamic) valoarea razei (R). Instrucțiunea din linia 40 ... De ce v-ați oprit? 40 r%=r. Nu sînteți obișnuiți cu variabila (r%) din stînga semnului egal al instrucțiunii de atribuire?

Amintiți-vă că în BASIC-AMSTRAD tipul unei variabile se poate stabili și prin caracterele speciale: "%"; " ! "; " $ " ce se atașează numelui

variabilei. Variabila r% este o variabilă numerică de tip întreg. Reţineţi aşadar, că r% şi r sînt variabile distincte, chiar dacă numele lor este acelaşi.

Instrucţiunea ZONE

Pentru afişarea unui rînd de spaţii am scris instrucţiunile 50, 100, 120. Instrucţiunile 55 şi 60 servesc la afişarea unui mic cap de tabel în poziţiile 1 şi 21 ale ecranului.

```
55 ZONE 20
60 PRINT "RAZA (Nr. real)", "RAZA (Nr. întreg)"
```

Instrucţiunea **PRINT** din linia 60 afişează cele două şiruri de caractere în poziţiile 1 şi 14 (tabulare standard pentru virgulă)! Ce ne facem? Tot BASIC-ul AMSTRAD "ne rezolvă", întrucît dispune de o instrucţiune specială – **ZONE** – care permite modificarea tabulării standard definite de separatorul virgulă într-o instrucţiune **PRINT**, cu un număr ce ia valori cuprinse între 1 şi 255.

Formatul general al instrucţiunii este:

Format general
⟨nr. linie⟩**ZONE**⟨nr. întreg⟩

unde ⟨nr. întreg⟩ ia valorile precizate mai sus.

Efectul instrucţiunii **ZONE** (linia 55) se răsfrînge şi asupra instrucţiunii **PRINT** din linia 70 care afişează valorile propriu-zise ale lui r şi r% în coloanele 1 şi 21.

Pentru calculul ariei şi volumului rezervorului sferic am scris instrucţiunile de atribuire 80 şi 90 în care variabila r este reală.

Rezultatul final este tipărit cu instrucţiunea 110 după un rînd de spaţii editat în linia 100.

Instrucţiunile **STOP** şi **END** termină execuţia programului redînd controlul sistemului.

PRINT SPC, PRINT TAB

Este momentul să vă prezentăm încă două opţiuni ale instrucţiunii **PRINT** şi anume: **SPC** şi **TAB**, pe care le vom utiliza foarte des în cadrul lucrării. **SPC** pregăteşte un număr întreg de spaţii libere indicat înainte de a imprima sau afişa articolul indicat, cu condiţia ca acesta din urmă să se menţină pe linia de imprimat.

Formatul general este:

Format general
⟨nr. linie⟩**PRINT**[# ⟨nr. canal⟩,][⟨listă⟩] [;][**SPC**(⟨nr. întreg⟩)][⟨listă⟩]

TAB realizează saltul la coloana al cărei număr este egal cu valoarea indicată.

Formatul general este:

Format general
⟨nr. linie⟩**PRINT**[# ⟨nr. canal⟩,][⟨listă⟩][;][**TAB**(⟨nr. intreg⟩)][⟨listă⟩]

Aplicaţie. Să se modifice instrucţiunea **PRINT** din liniile 60 şi 70 ale programului exemplu, utilizînd opţiunile **SPC** şi **TAB**. Noile instrucţiuni vor prelua funcţiunea instrucţiunii **ZONE** din linia 55 a programului.

```
60 print "RAZA (Nr. real)"; spc (7);
   "RAZA (Nr. întreg)"
70 print r; spc (15); r%
```

şi

```
60 print "RAZA (Nr. real)"; tab (21);
   "RAZA (Nr. întreg)"
70 print r; tab (21); r%
```

TEST

Simulaţi cu creionul şi hîrtia rezultatul execuţiei următoarelor programe:

a)
```
5 cls
10 a=8.25 : b=7.8 : a%=a
20 PRINT a, b
30 PRINT a, a%, b
40 ZONE 8
50 PRINT a, b, a%
60 PRINT "A", a, "A%", a%, "B", b
70 PRINT spc (0) "A"
80 PRINT spc (0); a%
90 PRINT tab (30); a; spc (5); "a"
```

b)
```
10 for i=0 to 6
20    print spc (i); "a"
30 next i
```

c)
```
5 a=1
10 for i=0 to 6
20    print spc (i); i
30 next i
```

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii şi corectării interactive a programelor BASIC-AMSTRAD sînt următoarele:

- linia program poate fi tastată numai după apariţia cursorului "■";
- pentru introducerea unui cuvînt cheie BASIC în calculator, puteţi utiliza indiferent MAJUSCULE sau minuscule;
- se preferă ca tastarea instrucţiunilor BASIC şi a programelor să se realizeze cu minuscule, deoarece la listarea programului (cu litere mari) cuvintele cheie prost ortografiate nu sînt convertite în majuscule;
- cînd se ajunge în marginea dreaptă a ecranului (s-au tastat mai mult de 40 caractere) caracterul următor trece automat în linia următoare;
- comanda **NEW** iniţializează sistemul BASIC ştergînd programul sau variabilele curente;
- punerea la zero a sistemului se realizează acţionînd simultan tastele **[CONTROL]**, **[SHIFT]**, **[ESC]**;
- comanda **LIST** cu formatul

Format general
LIST[⟨ansamblu de linii⟩][,⟨număr canal⟩]

listează programul pe canalul dorit (#0 este ecranul, #8 este imprimanta) cu majuscule;

Observație. Puteți omite primul sau ultimul număr de linie al parametrului ⟨ansamblu de linii⟩ în vederea listării programului de la început sau pînă la sfîrșit;

- listarea programului poate fi întreruptă prin acționarea o singură dată a tastei **[ESC]**. Pentru continuare, apăsați bara de spații. Dacă apăsați de două ori pe **[ESC]**, listarea programului încetează și se revine în modul direct;
- pentru ca linia să fie prelucrată se va acționa tasta **[RETURN]**. În condiții normale de funcționare a calculatorului, tasta **[ENTER]** are același efect ca și tasta **[RETURN]**, funcțiunile tastei **[ENTER]** putînd fi însă redefinite;
- pentru ștergerea unui caracter din stînga cursorului tastați [**DEL**];
- pentru tastarea majusculelor și a simbolurilor superioare înscrise pe taste mențineți apăsată tasta **[SHIFT]**;
- pentru tastarea majusculelor și a simbolurilor superioare înscrise pe taste puteți acționa, de asemenea, tastele **[CONTROL] [CAPS LOCK]** în această ordine. În acest caz, nu mai este nevoie să mențineți tot timpul apăsată tasta **[SHIFT]**;
- pentru ștergerea unui caracter care se găsește sub cursor tastați **[CLR]**;
- comanda **AUTO** al cărei format general este:

Format general
AUTO[⟨nr. linie⟩][,⟨increment⟩]

generează automat numerele de linie;

- parametrul facultativ ⟨nr. linie⟩ fixează primul număr de linie care se va genera. În absența acestuia se ia valoarea 10. Parametrul ⟨increment⟩, de asemenea facultativ, fixează pasul de secvențare a liniilor. Dacă nu se precizează, el ia valoarea 10. Pentru oprirea numerotării automate a liniilor acționați tasta [**RETURN**];
- dacă una din liniile programului a fost tastată incorect (vezi mesajul Syntax error sau alt mesaj de eroare) este posibilă editarea (modificarea) liniei fără a fi nevoiți să o retastăm;
- există trei metode pentru editarea unui program: **1) retastarea integrală a liniei; 2) editarea cu ajutorul cursorului; 3) COPY cursor sau copie cu ajutorul cursorului.**

Metoda de editare cu ajutorul cursorului (2) constă în poziționarea cursorului pe caracterul dorit, ștergerea acestuia și eventual înlocuirea lui cu un alt caracter. Procedura este următoarea: 1) tastați **edit** ⟨nr. linie⟩; 2) poziționați cursorul pe caracterul eronat; 3) tastați **[CLR]**, introduceți noul caracter; 4) tastați **[RETURN]**; 5) tastați **list [RETURN]** pentru a verifica dacă linia ⟨nr. linie⟩ este corectă.

Aplicație. Corectați cu metoda prezentată mai sus linia 10 a următorului program:

```
5 a = 3
10 plint a
20 end
```

Pentru a corecta linia 10 (l în r), tastați

```
edit 10 [RETURN]
```

Linia 10 apare sub linia 20, iar cursorul se poziționează pe litera p (primul caracter din plint). Pentru a elimina pe l din **plint,** acționați tasta [→] pînă ce cursorul se poziționează pe l. Tastați **[CLR]**. În acest moment a fost șters caracterul l. Tastați în locul lui l pe r și acționați tasta **[RETURN]**. Dați comanda **list [RETURN]** și veți obține:

```
5 a = 3
10 PRINT a
20 END
Ready
■
```

Observație. Dacă executați micuțul program de mai sus, veți obține pe ecranul monitorului (vezi fig. 3.5) — cum de altfel era și de așteptat — un mesaj de eroare și

imediat sub el linia 10 eronată. În acest caz aplicaţi metoda de editare cu ajutorul cursorului de la pasul 3.

```
run
Syntax error in 10
10 p[l]int a
```

Fig. 3.5.

Metoda COPY cursor (3) constă în copierea cu cel de-al doilea cursor **[SHIFT]**, şi una din tastele: [↑], [↓], [→], [←] a liniei de program pe care dorim să o modificăm. Procedura este următoarea: 1) menţineţi apăsată tasta **[SHIFT]** şi acţionaţi tasta cursorului [↑] pînă în momentul poziţionării cursorului la începutul liniei de modificat (cursorul principal rămîne imobilizat); 2) menţineţi apăsată tasta **[COPY]** pînă în momentul în care cursorul s-a plasat în poziţia dorită (linia pe care o corectaţi este rescrisă în acelaşi timp după ultima instrucţiune din program, iar cursorul principal se imobilizează în acelaşi loc ca şi cursorul de copiere; 3) tastaţi caracterul pe care doriţi să-l introduceţi; el apare numai în linia copiată (cursorul principal s-a deplasat în timp ce cursorul de copiere rămîne imobilizat); 4) menţineţi apăsată tasta **[COPY]** pentru copierea integrală a liniei asupra căreia s-au operat modificări. Tastaţi **[RETURN]** (cursorul de copiere dispare, iar cursorul principal se plasează sub linia copiată).

Aplicaţie. Corectaţi cu metoda **COPY CURSOR** linia 10 a programului din aplicaţia precedentă (pag. 98).

Pentru a corecta linia 10 (l în r) menţineţi apăsată tasta **[SHIFT]** şi acţionaţi tasta cursorului [↑] pînă cînd se suprapune cu cifra 1 din linia **10**. În fig. 3.6 (a) remarcaţi şi poziţia cursorului principal – aceeaşi!

```
5 a=3
[1]0 plint a
20 end
■
```

a)

```
Ready
5 a=3
10 p[l]int a
20 end
10 p■
```

b)

```
Ready
5  a=3
10 p█int a
20 end
10 pr■
```

c)

```
Ready
5  a=3
10 pl█nt  a
20 end
10 pr■
```

d)

```
Ready
5  a=3
10 plint  a■
20 end
10 print  a■
```

e)

```
Ready
5  a=3
10 plint  a
20 end
10 print a
■
```

f)

```
list
5  a=3
10 PRINT a
20 END
Ready
■
```

g)

Fig. 3.6 a–g.

În continuare menţineţi apăsată tasta [SHIFT] şi tastaţi COPY pînă cînd cursorul s-a suprapus peste litera l. În acelaşi timp, linia 10 este rescrisă la baza programului iar cursorul principal se imobilizează în acelaşi loc ca şi cursorul de copiere (vezi fig. 3.6, b).

Tastaţi r care apare numai pe linia 10 de la baza programului (vezi fig 3.6, c). Remarcaţi cum cursorul principal s-a deplasat, în timp ce cursorul de copiere a rămas pe loc. Acum apăsaţi pe [SHIFT] şi deplasaţi cursorul de copiere pînă se suprapune peste litera i (vezi fig. 3.6, d).

În sfirşit, menţineţi apăsată tasta [SHIFT] şi apăsaţi pe COPY în vederea copierii integrale a liniei 10 (vezi fig. 3.6, e).

Tastaţi [RETURN] astfel încît noua linie 10 (care o elimină pe precedenta linie 10) să fie introdusă în memoria calculatorului.

Observaţi cum cursorul de copiere dispare, iar cursorul principal se plasează sub linia 10 (vezi fig. 3.6 (f)).

Puteţi cere listarea programului tastînd:

list [RETURN]

Imaginea noului program este prezentată în fig. 3.6 (g).

Salvarea unui program BASIC-AMSTRAD pe dischetă*

Pentru început, trebuie să facem precizarea că AMSTRAD 6128 utilizează dischete de 3 inch (vezi fig. 3.7).

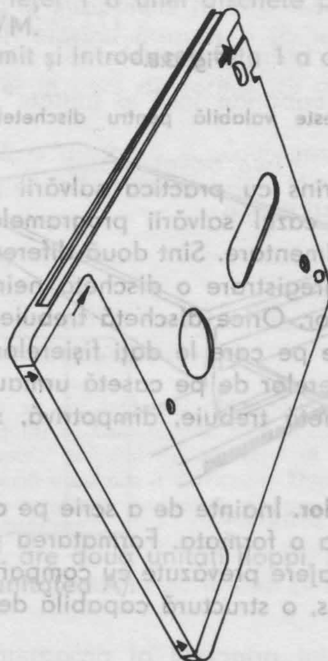


Fig. 3.7.

Fiecare faţă a dischetei poate fi utilizată separat: colţul din stînga al fiecărei feţe ce aparţine unei dischete poartă o săgeată ce indică o

* Se pot utiliza, de asemenea, şi casetele magnetice.

gaură acoperită de un obturator. Aceasta este o gaură de protecţie la scriere, care va împiedica să se scrie peste informaţiile înregistrate. Cînd gaura este închisă, calculatorul poate scrie pe dischetă. Cînd este deschisă, nu poate fi scris nimic. Pentru închiderea şi deschiderea găurii de protecţie procedaţi ca în fig. 3.8.

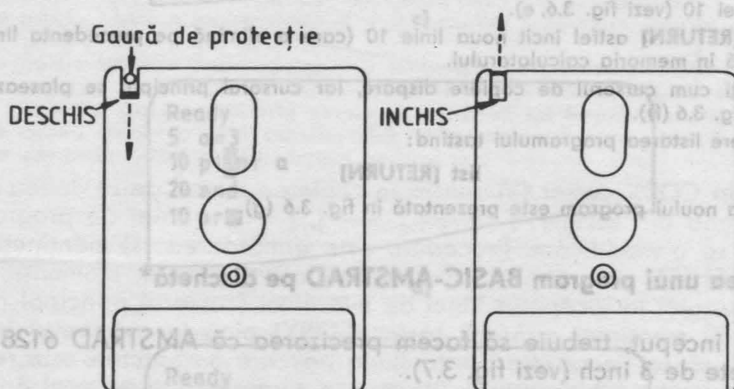


Fig. 3.8.

Observaţie. Procedura este valabilă pentru dischetele CF-2 (Compact Floppy) AMSOFT.

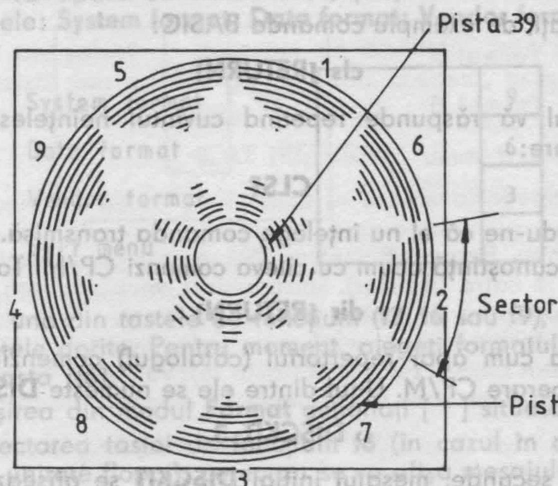
Deşi dvs. v-aţi deprins cu practica salvării programelor BASIC pe casetă, va trebui ca, în cazul salvării programelor pe dischetă, să vă însuşiţi cîteva noţiuni suplimentare. Sînt două diferenţe esenţiale. Nu puteţi utiliza niciodată pentru înregistrare o dischetă neiniţializată (virgină) cum procedaţi în cazul casetelor. Orice dischetă trebuie să fie mai întîi formatată. Va trebui ca numele pe care le daţi fişierelor de pe dischetă să fie corecte. Dacă numele fişierelor de pe casetă urmau reguli nu întotdeauna precise, cele de pe dischetă trebuie, dimpotrivă, să se conformeze strict regulilor CP/M.

Formatarea dischetelor. Inainte de a scrie pe o dischetă neiniţializată, trebuie să începem prin a o formata. Formatarea poate fi înţeleasă ca şi construirea unui set de etajere prevăzute cu compartimente pentru stocarea informaţiei sau, altfel spus, o structură capabilă de a primi şi totodată de a restitui informaţii.

Operaţia de formatare împarte fiecare faţă a dischetei în 360 de zone distincte.

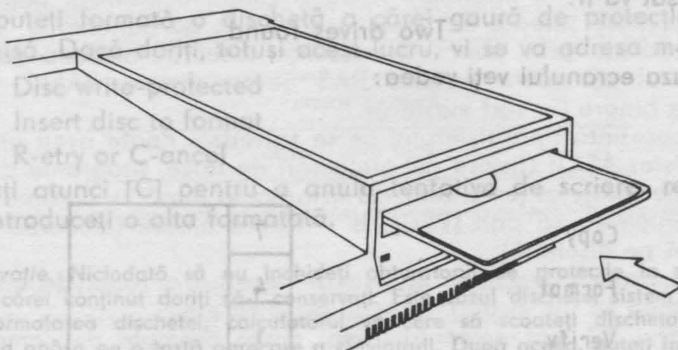
Suprafaţa unei dischete este împărţită în 40 de piste concentrice (0–39), fiecare pistă fiind împărţită în 9 sectoare.

Un sector, indiferent de pistă, poate conţine 512 octeţi sau 1',0 Ko spaţiu disponibil pe fiecare faţă a dischetei.



Cum folosim dischetele CP/M? Comenzi CP/M. Înainte de a utiliza o dischetă neinițializată în vederea scrierii și citirii programelor BASIC, formatați-o cu ajutorul feței 1 a unei dischete pe care se află înregistrat sistemul de operare CP/M.

Treziți geniul adormit și introduceți fața 1 a dischetei CP/M în unitatea floppi:



Dacă sistemul dvs. are două unități floppi, introduceți discheta CP/M în unitatea principală (unitatea A).

Tastați:

| cpm **[RETURN]**

Observație. Bara verticală | se obține prin acționarea tastelor **[SHIFT]** și @.

După cîteva secunde, pe ecran va apare mesajul:

CP/M plus AMSTRAD Consumer Electronics plc

care ne atenționează că sistemul este sub controlul CP/M plus.

Simbolul **A>** de pe ecran este echivalentul lui **Ready** sub BASIC. El ne avertizează că sistemul este pregătit să primească comenzile dvs.

Dacă tastaţi, de exemplu comanda BASIC:

cls [RETURN]

. . . calculatorul vă răspunde repetînd cuvîntul neînţeles, urmat de un semn de întrebare:

CLS?

. . . demonstrîndu-ne că el nu înţelege comanda transmisă.

Să facem cunoştinţă acum cu cîteva comenzi CP/M. Tastaţi:

dir [RETURN]

Veţi vedea cum apar repertoriul (catalogul) comenzilor şi utilitarele sistemului de operare CP/M. Unul dintre ele se numeşte **DISCKIT 3**. Tastaţi:

DISCKIT 3

Timp de cîteva secunde, mesajul iniţial **DISCKIT** se afişează în partea de sus a ecranului, urmat de:

One drive found

indicînd că utilitarul **DISCKIT** este în exploatare şi că AMSTRAD-ul a recunoscut utilizarea sistemului cu o singură unitate floppi (cea care este integrată în calculator).

Dacă sistemul dvs. cuprinde încă o unitate suplimentară de floppi, mesajul afişat va fi:

Two drives found

La baza ecranului veţi vedea:

Copy	7
Format	4
Verify	1
Exit from program	0

Este vorba de meniul principal al programului **DISCKIT**. Numerele înscrise în interiorul căsuţelor corespund tastelor de funcţiuni situate la dreapta claviaturii (**f0, f1, f4** şi **f7**). Selecţia o puteţi face cu una din tastele precizate. Acţionaţi tasta [**f0**] pentru a ieşi din programul **DISCKIT**.

Deoarece noi dorim formatarea unei dischete, acţionaţi tasta [**f4**].

Remarcă. Formatarea unei dischete deja înregistrate are ca efect distrugerea vechiului conţinut.

Pe ecran va apare un nou meniu, propunîndu-vă să optaţi pentru unul din formatele: **System format; Data format; Vendor format.**

System format	9
Data format	6
Vendor format	3
Exit menu	

Acţionînd una din tastele de funcţiuni (**f3**, **f6** sau **f9**), puteţi selecţiona unul din formatele dorite. Pentru moment, alegeţi formatul **Data**, acţionînd în consecinţă tasta [**f6**].

Pentru ieşirea din modul **Format** acţionaţi [·] situată sub **f3**, **f6** şi **f9**. După selectarea tastei de funcţiuni **f6** (în cazul în care sistemul nu posedă decît o unitate floppi), pe ecran se va afişa mesajul:

[Y] Format as Data
Any other key to exit menu

În acest moment, scoateţi discheta CP/M, introduceţi discheta pe care doriţi să o formataţi şi tastaţi [Y].

Formatarea dischetei se efectuează pistă cu pistă (de la pista 0 la pista 39), numărul pistei în curs de formatare afişîndu-se în colţul de sus din stînga ecranului.

Nu puteţi formată o dischetă a cărei gaură de protecţie la scriere este deschisă. Dacă doriţi, totuşi acest lucru, vi se va adresa mesajul:

Disc write-protected
Insert disc to format
R-etry or C-ancel

Tastaţi atunci [C] pentru a anula tentativa de scriere, retrageţi discheta şi introduceţi o alta formatată.

Observaţie. Niciodată să nu închideţi obturatorul de protecţie la scriere a unei dischete al cărei conţinut doriţi să-l conservaţi. Este cazul dischetei sistem CP/M. Odată terminată formatarea dischetei, calculatorul vă cere să scoateţi discheta din unitatea floppi şi de a apăsa pe o tastă oarecare a claviaturii. După aceea, puteţi introduce o altă dischetă pe care doriţi să o formataţi şi apăsaţi din nou pe [Y] pentru a reveni în meniul principal al utilitarului **DISCKIT.**

Pentru reiniţializarea sistemului, tastaţi **[CONTROL]**, **[SHIFT]** şi **[ESC]** în această ordine.

Observaţie. Păstraţi întotdeauna o copie a dischetei CP/M.

Formatarea dischetelor într-un calculator cu două unităţi floppi. Urmaţi instrucţiunile prezentate mai sus, selectaţi opţiunea **Format** din meniul principal al programului utilitar **DISCKIT,** acţionînd tasta de funcţiuni [**f4**] şi apoi selectaţi **Data format** cu ajutorul tastei de funcţiuni [**f6**].

Veţi vedea cum, pe ecran, se afişează al treilea meniu de opţiuni:

Format A:	8
Format B:	5
Exit menu	2

Optînd pentru **Format B** (tasta [f5]) puteţi lăsa discheta sistem în unitatea A (faţa 1 către exterior), introducînd deci discheta pe care doriţi s-o formataţi în unitatea B. Tastaţi aşadar [Y] pentru formatare sau oricare altă tastă pentru a reveni în meniul principal al utilitarului **DISCKIT**.

Dacă aţi optat pentru **Format A** (tasta **f8**), va trebui să scoateţi discheta sistem din unitatea A pe care s-o înlocuiţi cu discheta de formatat.

Observaţie. Nu uitaţi să protejaţi discheta sistem împotriva unei distrugeri accidentale a informaţiilor pe care le conţine.

Salvarea unui program pe dischetă se realizează cu comanda **SAVE** al cărei format general este:

SAVE⟨nume fişier⟩[,⟨tip fişier⟩][,⟨parametrii binari⟩]

unde, ⟨nume fişier⟩ se compune din două părţi sau zone. Prima, obligatorie, poate să conţină pînă la 8 caractere, litere sau cifre (spaţiul şi semnele de punctuaţie sînt interzise). A doua zonă, facultativă, poate conţine pînă la 3 caractere diferite de spaţiu sau semne de punctuaţie. Cele două zone sînt separate prin punct. În absenţa celei de-a doua zone specificate, sistemul consideră automat.**BAS** pentru fişiere BASIC sau ·**BIN** pentru fişiere binare (în cod maşină);

⟨tip fişier⟩ reprezintă tipul fişierului ce se salvează. Poate avea una din valorile: a – fişier ASCII (fişiere de prelucrări de texte, programe de aplicaţii etc.); p – fişier protejat (este interzisă listarea sa după încărcarea în memorie sau întreruperea sa prin **[ESC]**; b – fişiere binare (stocarea blocurilor de date RAM pe dischetă);

⟨parametrii binari⟩ reprezintă: adresa de început a zonei de memorie de salvat, lungimea sa (nr. octeţi), punctul de intrare etc.

În tabelul 3.1 sînt prezentate mai multe exemple de comenzi **SAVE** comentate.

Tabelul 3.1

Comandă	Funcţiune
SAVE "prog.1"	salvează fişierul în mod BASIC neprotejat
SAVE "prog.1",P	salvează fişierul în mod BASIC protejat
SAVE "prog.1",A	salvează fişierul în mod ASCII
SAVE "prog.1",B	salvează fişierul în mod binar
SAVE "pag. ecran", b, 49152, 16384	salvează conţinutul ecranului (vidaj ecran) de la adresa de început a memoriei ecran – 49152 pe lungimea de 16384 memorie ecran.

Listarea catalogului (repertoriului) unei dischete

După ce aţi salvat un program pe dischetă, este bine să verificaţi dacă acesta există într-adevăr fizic. Pentru aceasta, este necesar să citiţi numai **CAT**alogul dischetei respective. Tastaţi:

CAT [RETURN]

şi veţi vedea cum pe ecran se listează în ordine alfabetică numele tuturor fişierelor prezente, precum şi lungimea programului (arondată la Ko superior). Totodată se afişează identificatorul dischetei şi al utilizatorului, precum şi numărul de octeţi disponibili.

Aplicaţie. Salvaţi programul exemplu pe o dischetă (neiniţializată) sub numele PC3AMST. Verificaţi dacă programul a fost salvat.

Încărcarea unui program de pe dischetă

Programele de pe dischetă sînt încărcate în memoria calculatorului cu ajutorul comenzii **load** al cărei format general este:

Format general
LOAD⟨nume fişier⟩ [,⟨adresă⟩]

Cu opţiunea de adresă se încarcă un fişier binar la adresa indicată în locul adresei în care el se găsea în momentul salvării. Pentru un demaraj rapid se foloseşte comanda **run** al cărei format general este:

Format general
RUN⟨nume fişier⟩

Observaţie. Comanda **run** este singura comandă care permite execuţia programelor protejate.

În tabelul 3.2 sînt prezentate două exemple de încărcare a unui program de pe dischetă în memoria calculatorului.

Tabelul 3.2

Comandă	Funcţiune
load "prog.1" **[RETURN]** **run [RETURN]**	încarcă în memorie de pe dischetă ⟨prog.1⟩
run "prog.1" **[RETURN]**	idem, dar încărcarea este mai rapidă

TEST

Indicaţi o procedură de copiere a unui program BASIC de pe o dischetă pe o altă dischetă.

Dacă dispuneţi de un sistem cu două dischete, puteţi introduce discheta cu programul de încărcat în unitatea B de exemplu, iar discheta care primeşte programul în unitatea A.

```
| b [RETURN]
load "prog.1" [RETURN]
| a [RETURN]
save "prog.1" [RETURN]
```

Aplicaţie. Încărcaţi PC3AMST în memoria calculatorului cu una din comenzile studiate (vezi aplicaţia de la pag. 107).

Remarci

- Nu uitaţi niciodată să scoateţi discheta din unitatea de floppi, înainte de a pune calculatorul sub tensiune sau după ce l-aţi oprit. Riscaţi pierderea programelor şi a datelor înregistrate.
- Faceţi întotdeauna o copie a dischetelor care conţin programe preţioase. Este cazul dischetei CP/M furnizată odată cu calculatorul AMSTRAD.
- Verificaţi întotdeauna sistemul de protecţie la scriere a dischetei pe care se află sistemul de operare CP/M.
- Nu atingeţi suprafaţa magnetică a dischetei.
- Nu extrageţi niciodată discheta în timpul citirii sau scrierii.
- Amintiţi-vă că formatarea unei dischete presupune ştergerea vechiului conţinut.

Execuţia programului. Reguli generale

Pentru lansarea în execuţie a unui program BASIC-AMSTRAD, trebuie cunoscute şi respectate următoarele **reguli** generale:

- comanda **RUN** realizează lansarea în execuţie a programului curent existent în memoria calculatorului de la instrucţiunea cu cel mai mic număr de linie;
- comanda **RUN** ⟨nr. linie⟩ lansează în execuţie programul curent existent în memoria calculatorului de la instrucţiunea care are numărul de linie egal cu ⟨nr. linie⟩;
- comanda **RUN** ⟨şir alfanumeric⟩ încarcă şi execută un program BASIC sau un program obiect înregistrat pe dischetă. Această comandă permite accesul direct la programele BASIC protejate;
- instrucţiunea **STOP** întrerupe execuţia unui program. Pentru reluarea execuţiei programului tastaţi **CONT**;
- întreruperea momentană a execuţiei unui program se realizează prin acţionarea tastei **[ESC]** o singură dată; reluarea execuţiei se face prin apăsarea unei alte taste iar oprirea completă a programului se realizează tastînd încă o dată **[ESC]**;
- după introducerea datelor prin instrucţiunea **INPUT** se tastează **[RETURN]**;
- lista mesajelor de eroare se găseşte în vol. 2, pag. 5.

Dacă programul exemplu nu se află în memoria calculatorului, vă invităm să-l încărcaţi de pe dischetă (vezi aplicaţia de la pag. 108 pe care trebuia să o realizaţi singuri) ca după aceea să-l executaţi. Să urmărim împreună execuţia programului.

Pe ecran a apărut "Raza=?", calculatorul aşteptînd ca dvs. să introduceţi valoarea pe care o doriţi. Vă propunem ca, de această dată, să tastaţi 3.3, **[RETURN]**, adică un număr real. Între timp s-a afişat şi capul de tabel sub care s-au imprimat (sub controlul aceleiaşi instrucţiuni **Zone**) valorile 3.3 (*număr real*) şi 3 (*număr întreg*).

După ce a calculat aria (a) şi volumul (v) rezervorului sferic de rază r (număr real), calculatorul AMSTRAD afişează rezultatul ca în figura:

```
Raza =? 3.3
RAZA (Nr. real)                 RAZA(Nr. întreg)
 3.3                             3
R=3.3  A=136.847776             V=150.532554
```

Puteţi relua execuţia programului ori de cîte ori doriţi; încercaţi să introduceţi pentru r şi alte valori ca: 3, 0.8 etc.

Aplicaţie. Modificaţi programul exemplu, astfel încît să calculeze aria şi volumul rezervorului sferic numai pentru valori întregi ale razei r.

TESTE

a) Se consideră următorul program BASIC:

```
10 REM PROGRAMUL CITEŞTE TREI
20 REM NOTE (N1, N2, N3) ALE UNUI
30 REM STUDENT, CALCULEAZĂ
   MEDIA (M)
40 REM ŞI TIPĂREŞTE NOTELE ŞI MEDIA
50 PRINT "INTRODUCEŢI PRIMA NOTĂ";
60 INPUT N1
70 PRINT "INTRODUCEŢI A DOUA
   NOTĂ";
90 PRINT "INTRODUCEŢI A TREIA
   NOTĂ";
100 INPUT N3
110 LET M=(N1+N2+N3)/3
120 PRINT
130 PRINT "MEDIA ESTE"; M
140 END
75 INPUT N2
```

În fig. 3.9 se prezintă dialogul cu calculatorul AMSTRAD.

```
RUN
INTRODUCETI PRIMA NOTA ? 8
INTRODUCETI A DOUA NOTA ? 0
INTRODUCETI A TREIA NOTA ?10
MEDIA ESTE 6
```

Fig. 3.9

După fiecare semn de întrebare utilizatorul tastează valoarea variabilei apoi apasă pe tasta [**RETURN**].

După ce au fost introduse cele trei valori, calculatorul calculează media aritmetică şi afişează rezultatul.

Ce valoare a introdus utilizatorul pentru variabila N1?

Dar pentru variabila N2?

```
RUN
INTRODUCEŢI PRIMA NOTĂ? 5
INTRODUCEŢI A DOUA NOTĂ? 0
INTRODUCEŢI A TREIA NOTĂ? 10
MEDIA ESTE 5
```

b) Scrieţi un program BASIC-AMSTRAD pentru înmulţirea a două binoame:

(AX+B) · (CX+D).

Valorile lui A, B, C, D se vor introduce dinamic. Pe ecran se vor afişa trei valori reprezentînd coeficienţii lui X^2, X şi termenul liber.

```
10 PRINT "A, B, C, D";
20 INPUT A, B, C, D
30 PRINT A * C; A * D+B * C;
   B * D
40 END
run
A, B, C, D ? 1, 2, 3, 4
3  10  8
Ready
■
```

c) Precizaţi rezultatul execuţiei următorului program BASIC-AMSTRAD dacă lui A şi B li se atribuie dinamic valorile 3, respectiv 2.

```
10 cls
20 print "cine este 'A' "
30 input a
40 print "cine este 'B' "
50 input b
```

```
60 cls
70 print a"+"b"="a+b
80 end
```

R.

```
3+2=5
Ready
```



☐ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 203

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-COMMODORE trebuie cunoscute şi respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- o linie program poate conţine una sau mai multe instrucţiuni separate între ele prin două puncte (" : ");
- o variabilă este identificată printr-un şir de caractere care începe cu o literă, urmată de un număr oarecare de litere sau cifre dintre care numai primele două caractere sînt semnificative;
- variabilele admise sînt de următoarele tipuri: numerice reale, numerice întregi, şiruri de caractere;
- tipul unei variabile se poate stabili şi prin caracterele speciale: "%" (pentru variabilele numerice întregi); "$" (pentru şiruri de caractere) ce se ataşează numelui variabilei;
- numele unei variabile indexate este orice nume de variabilă acceptat de către BASIC-COMMODORE;
- valoarea minimă a indicelui unui tablou este zero;
- numărul de indici ai unei variabile indexate este de la 1–3;
- operatorii aritmetici admişi sînt: "+" (pentru adunare); "–" (pentru scădere); " * " (pentru înmulţire); "/" (pentru împărţire); "↑" (pentru ridicare la putere);
- operatorii de relaţie admişi sînt: "="; "<"; ">"; "<="; "=>";
- operatorii logici admişi sînt: **NOT, AND, OR.**
- funcţiile matematice admise sînt: **COS, EXP, LOG, RND, SIN, SQR, TAN, ATN, INT, SGN;**
- funcţiile admise pentru şirurile de caractere sînt: **LEN, POS, MID$, LEFT$, RIGHT$;**
- funcţiile de conversie admise sînt: **CHR$, VAL, STR$, ASC;**
- alte funcţii (auxiliare) disponibile sînt: **TAB, SPC, POS, FRE, USR, SYS, WAIT;**
- instrucţiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN, ON ... GOTO;**
- instrucţiunea de ciclare permisă este **FOR ... NEXT;**
- instrucţiunile de intrare/ieşire permise sînt: **READ, DATA, RESTORE, INPUT, GET, PRINT, POKE; PEEK;**
- instrucţiunile admise pentru operaţiile de intrare/ieşire pe disc sînt: **OPEN, CLOSE, PRINT, INPUT, GET, SAVE, LOAD;**
- limbajul nu are instrucţiuni grafice speciale. Se pot utiliza instrucţiunile **POKE** şi **PEEK;**
- limbajul nu are instrucţiuni speciale pentru producerea sunetelor. Se pot utiliza instrucţiunile **POKE** şi **PEEK.**

Aplicaţi aceste reguli la scrierea programului. Vă puteţi ghida şi după lista programului precedent. Pentru tabularea cerută la proiectarea programului puteţi folosi una din instrucţiunile **PRINT TAB** sau **PRINT SPC** (identice cu cele de la calculatorul AMSTRAD). Ca şi în programul precedent puteţi

folosi variabile întregi şi reale denumite cu acelaşi nume (numai primele două caractere sînt semnificative!).

La tipărirea rezultatului final, nu uitaţi să editaţi şi valoarea razei pe care o citiţi cu instrucţiunea **INPUT**.

Introducerea programului. Reguli generale

Regulile care stau la baza introducerii şi corectării interactive a programelor BASIC-COMMODORE sînt:

- linia program poate fi tastată numai după apariţia cursorului "■";
- comanda **LIST** tipăreşte o parte sau tot programul aflat în memorie (vezi comanda AMSTRAD);
- comanda **NEW** elimină din memoria calculatorului programul rezident;
- comanda **CLR** iniţializează valorile tuturor variabilelor.

Salvarea programelor pe un suport extern de memorie. Un program BASIC-COMMODORE poate fi salvat pe casetă sau dischetă.

Formatul general al comenzii este:

Format general
SAVE["⟨nume fişier⟩"][,a][,b]

Dacă **SAVE** se execută fără parametri, programul curent din memorie (RAM) este înregistrat pe casetă. Dacă se indică ⟨nume fişier⟩ programul curent se salvează (pe casetă) sub acel nume. Dacă salvarea se face pe dischetă, ⟨nume fişier⟩ este obligatoriu. Cel de-al doilea parametru indică perifericul. Dacă se indică 8, programul ⟨nume fişier⟩ se salvează pe dischetă. Dacă cel de-al doilea parametru (b) nu este folosit sau are valoarea 1, fişierul va fi înregistrat pe casetă. Dacă parametrul b are valoarea 1, atunci se mai poate indica şi un al treilea parametru. Dacă şi pentru cel de-al treilea parametru se fixează valoarea 1, la sfîrşitul fişierului se va imprima eticheta de End of Tape.

În tabelul 3.3 se prezintă cîteva exemple de utilizare a comenzii.

Tabelul 3.3

Comandă	Funcţiune
SAVE "EX1"	salvează un program pe casetă, sub numele EX1.
SAVE "EX1", 8	salvează un program pe dischetă, sub numele EX1.
SAVE "EX1",1,1	salvează un program pe casetă, sub numele EX1, cu imprimarea unei etichete End of Tape.

Observaţie. Programele salvate cu **SAVE** se încarcă cu comanda **LOAD**.

Aplicaţie. Salvaţi pe o dischetă (iniţializată), sub numele PC3COM, programul exemplu.

Încărcarea programelor de pe un suport extern

Un program BASIC-COMMODORE poate fi încărcat în memoria calculatorului de pe un suport extern (casetă, dischetă) cu comanda **LOAD**.

Formatul general al comenzii este:

Format general
LOAD ["⟨nume program⟩"] [,a]

Parametrul "a" specifică tipul perifericului (casetă/dischetă). Dacă programul urmează a fi încărcat de pe o dischetă, a ia valoarea 8. De menționat că ⟨nume program⟩ este necesar numai pentru programele salvate pe dischetă. Dacă în loc de ⟨nume program⟩ se indică asterisc "*", de pe dischetă se va încărca (în memoria calculatorului) numai primul program.

În tabelul 3.4 se prezintă mai multe exemple de utilizare a comenzii **LOAD**.

Tabelul 3.4

Comandă	Funcțiune
LOAD	încarcă de pe casetă următorul program.
LOAD "EX1"	încarcă de pe casetă programul EX1.
LOAD "*",8	încarcă de pe dischetă primul program găsit.
LOAD "EX1",8	încarcă de pe dischetă programul EX1.

Remarci

- Cînd se execută **LOAD** pentru un program înregistrat pe casetă se afișează mesajele:

  PRESS PLAY ON TAPE
  SEARCHING FOR nume program
  FOUND nume program

- Odată cu încărcarea programului în memoria calculatorului se afișează mesajul:

  LOADING

urmat de:

OK
READY

atunci cînd procedura de încărcare **LOAD** este completă.

Aplicație. Încărcați în memoria RAM a calculatorului programul P3COM înregistrat pe dischetă.

Pentru verificarea modului în care a fost realizată salvarea unui program se utilizează comanda **VERIFY** al cărei format general este:

Format general
VERIFY [⟨nume program⟩ , ⟨dispozitiv⟩]

Dacă în urma verificării celor două programe (din memorie și de pe suportul extern) rezultă că cele două programe sînt identice, se afișează mesajul:

OK
READY

în caz contrar, apare mesajul:

? VERIFY
READY

Execuția programului.

Reguli generale

Pentru lansarea în execuție a unui program BASIC-COMMODORE trebuie cunoscute și respectate următoarele **reguli** generale:

- Comanda **RUN** al cărei format general este:

Format general
RUN [⟨nr. linie⟩]

lansează în execuție programul curent din memoria internă a calculatorului, de la prima linie program (în absența parametrului ⟨nr. linie⟩) sau de la cea cu numărul de linie egal cu ⟨nr. linie⟩;

Observație. Dacă într-o comandă **RUN** se specifică ⟨nr. linie⟩ dar linia cu numărul respectiv nu există în program, pe ecran se afișează mesajul:

? UNDEF'D STATEMENT ERROR

- la întîlnirea instrucțiunii **STOP**, execuția programului se oprește temporar cu mesajul:
  BREAK IN XXX

  unde XXX este numărul de linie al instrucțiunii **STOP**. Pentru reluarea execuției tastați **[CONT]**;
- execuția unui program poate fi oprită prin acționarea tastei **[BREAK]**.

Din acest moment, nimeni nu vă mai oprește să conversați cu COMMODORE. Răspundeți concret la întrebarea ce vi se adresează ("RAZA=?") și analizați răspunsul primit. Oricare ar fi rezultatul, continuați!

Aplicație. Să se determine coordonatele centrului de greutate al plăcii din figura de mai jos, dacă dimensiunile sînt următoarele: AB=20 cm; BD=24 cm; ED=10 cm; AN=2 cm; NL=18 cm; LK=20 cm; FK=8 cm.

Indicație. Se împarte placa în trei arii elementare și se alege sistemul de axe din figură.

Specificațiile de programare și documentația de proiectare sînt prezentate în modulele de analiză și proiectare structurată, fig. 3.10.

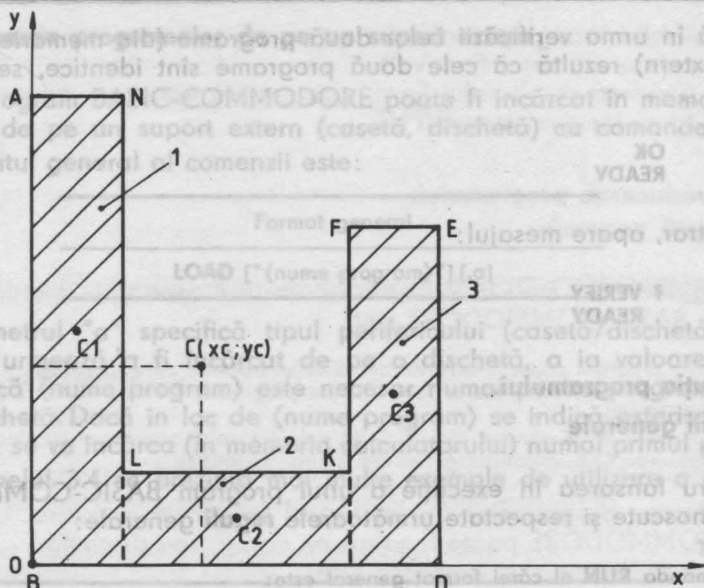


TABELA DE VARIABILE

Variabile de intrare	Variabila de stare	Variabile de ieșire
A1 : aria primului corp A2 : aria celui de-al doilea corp A3 : aria celui de-al treilea corp X1 : abscisa centrului de greutate al primului corp X2 : abscisa centrului de greutate al celui de-al doilea corp X3 : abscisa centrului de greutate al celui de-al treilea corp Y1 : ordonata centrului de greutate al primului corp Y2 : ordonata centrului de greutate al celui de-al doilea corp Y3 : ordonata centrului de greutate al celui de-al treilea corp	A : aria totală	X : abscisa centrului de greutate Y : ordonata centrului de greutate

a)

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal ariile, abscisele și ordonatele centrelor de greutate ale celor trei corpuri, calculează și afișează coordonatele centrului de greutate al plăcii.

Intrări

Ariile, abscisele și ordonatele centrelor de greutate ale celor trei corpuri care compun placa.

Ieșiri

Abscisa și ordonata centrului de greutate al plăcii.

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează abscisa centrului de greutate.
3. Calculează ordonata centrului de greutate.
4. Afișează rezultatul.
5. Stop.

b)

Fig. 3.10. Modulele de analiză și proiectare structurată: *a)* tabela de variabile; *b)* specificații de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

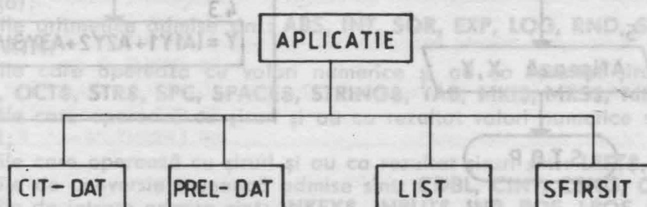
Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2,3
LIST	4
SFÎRȘIT	5

c)

DATELE DE INTRARE

A1	A2	A3	X1	X2	X3	Y1	Y2	Y3
40	40	20	1	12	23	10	1	5

d)



e)

PSEUDOCODUL

```
APLICAȚIE   SEQ
CIT-DAT     SEQ
            INPUT A1, A2, A3
            INPUT X1, X2, X3
            INPUT Y1, Y2, Y3
CIT-DAT     END
PREL-DAT    SEQ
            A=A1+A2+A3
            X=(A1*X1+A2*X2+A3*X3)/A
            Y=(A1*Y1+A2*Y2+A3*Y3)/A
PREL-DAT    END
            PRINT "X="; X; "Y="; Y
APLICAȚIE   END
```

f)

Fig. 3.10 *c)* alocarea funcțiunilor de prelucrare; *d)* datele de intrare; *e)* diagrama de structură; *f)* pseudocodul;

```
10 REM PROGRAMUL CALCULEAZĂ CO
   ORDONATELE
20 REM CENTRULUI DE GREUTATE AL
30 REM UNEI PLĂCI PLANE
40 PRINT "A1, A2, A3";
50 INPUT A1, A2, A3;
60 PRINT "X1, X2, X3";
70 INPUT X1, X2, X3
80 PRINT "Y1, Y2, Y3";
90 INPUT Y1, Y2, Y3
100 A=A1+A2+A3
110 X=(A1*X1+A2*X2+A3*X3)/A
120 Y=(A1*Y1+A2*Y2+A3*Y3)/A
125 PRINT
130 PRINT "X="; X, "Y="; Y
140 END
150 STOP
RUN
A1, A2, A3 ?
X1, X2, X3 ?
Y1, Y2, Y3 ?
X=9.80          Y=5.40
Ready
```

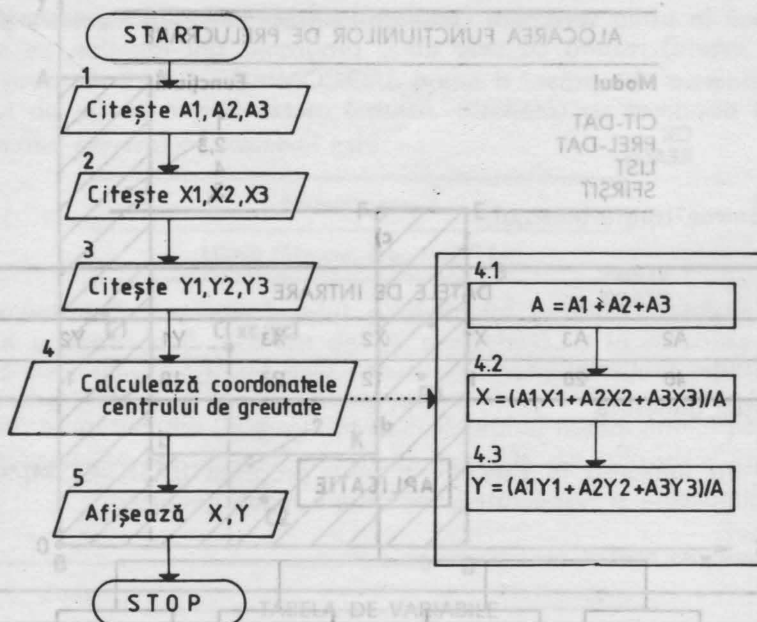


Fig. 3.10. *g)* schema logică.

TEST

Două conductoare foarte lungi, paralele, aflate la distanţa d=10 cm unul de celălalt, sînt parcurse de curenţi de acelaşi sens, de intensităţi I_1=5A şi I_2=10A. Să se calculeze inducţia magnetică a cîmpului rezultant într-un punct situat la jumătatea distanţei dintre cele două conductoare.

Indicaţie. Se va utiliza formula de calcul $B=\frac{\mu_0}{\pi d}(I_2-I_1)$. Valorile curenţilor şi distanţei dintre conductori se vor citi dinamic. Constanta de permeabilitate magnetică a vidului (μ_0) are valoarea: $\mu_0=4\pi\cdot10^{-7}N/A^2$.

```
5 PRINT "D, I2, I1";
10 INPUT D, I2, I1
20 MIU=(4*PI)/10↑7
30 B=(MIU/PI*D)*(I2-I1)
35 PRINT
40 PRINT "INDUCŢIA MAGNETICĂ="; B; "T"
50 END
RUN
D, I2, I1? 0.1, 10,5
INDUCŢIA MAGNETICĂ=
```

□ Codificarea în limbajul BASIC-80

vol. 2, pag. 203

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-80 trebuie cunoscute şi respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- numărul liniei este cuprins între 0 şi 65529;

- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- numele variabilelor numerice sînt formate dintr-un număr nelimitat de caractere (litere, cifre, caracterul punct) care încep cu o literă, dar numai primele 40 de caractere sînt semnificative;
- variabilele BASIC-80 pot fi: șir, întregi, simplă precizie sau dublă precizie;
- tipul unei variabile se poate stabili și prin caracterele speciale: "$" (variabilă de tip șir); "%" (variabilă întreagă); "!" (variabilă simplă precizie); "#" (variabilă dublă precizie) ce se atașează numelui variabilei;
- o variabilă poate fi declarată și explicit cu ajutorul instrucțiunilor **DEFINT, DEFSTR, DEFSNG, DEFDBL**;
- numele unei variabile indexate este orice nume de variabilă acceptat de BASIC-80;
- numărul maxim pentru dimensiunile unei variabile indexate este 255;
- numărul maxim de elemente/dimensiune este 32767;
- operatorii aritmetici admiși sînt: "+", "−", "*", "/", "↑";
- operatorii de relație admiși sînt: "=", "<", ">", "<=", "=<", "=>", ">=", "<>", "><".;
- operatorii logici admiși sînt: **NOT, AND, OR, XOR, IMP** (implicație), **EQV** (echivalență);
- funcțiile aritmetice admise sînt: **ABS, INT, SQR, EXP, LOG, RND, SGN, COS, SIN, TAG, ATN**;
- funcțiile care operează cu valori numerice și au ca rezultat șiruri sînt: **CHR$, HEX$, OCT$, STR$, SPC, SPACE$, STRING$, TAB, MKI$, MKS$, MKD$**;
- funcțiile care operează cu șiruri și au ca rezultat valori numerice sînt: **ASC, LEN, INSTR**;
- funcțiile care operează cu șiruri și au ca rezultat șiruri sînt: **LEFT$, RIGHT, MID$**;
- funcțiile de conversie numerică admise sînt: **CDBL, CINT, CSNG, CVI, CVS, CVD**;
- funcțiile de intrare admise sînt: **INKEY$, INPUT$, INP, POS, LPOS, PEEK**;
- alte funcții disponibile sînt: **FREE, USR, VARPTR, LOC, EOF, SWAP**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN ... ELSE, IF ... GOTO ..., ON ... GOTO, ON ERROR GOTO**;
- instrucțiunile de ciclare permise sînt: **FOR ... NEXT, WHILE ... WEND**;
- instrucțiunile de intrare/ieșire permise sînt: **INPUT, LINE INPUT, PRINT, PRINT USING, LPRINT, LPRINT USING, WRITE, READ, DATA, RESTORE, OUT, POKE**;
- instrucțiunile pentru operațiile de intrare/ieșire pe disc sînt: **SAVE, LOAD, OPEN, CLOSE, INPUT, LINE INPUT, PRINT, PRINT USING, WRITE, GET, PUT, KILL, NAME**;
- comenzile suplimentare în BASIC GRAPHIC (GBASIC) admise sînt: **VIEWPORT, WINDOW, MOVE, RMOVE, DRAW, RDRAW, ROTATE.**

Vom aplica aceste reguli la scrierea programului.

```
 5 PRINT CHR$ (24)
10 REM Programul citeşte raza (R)
15 REM a unui rezervor sferic
20 REM calculează aria (A),
   volumul (V) şi
30 REM tipăreşte: Raza, Aria,
   Volumul
35 PI=3.14159
40 PRINT "Raza=";
45 INPUT R
50 R%=R
60 PRINT "RAZA (Nr. real)"; SPC
   (7); "RAZA (Nr. întreg)
65 PRINT
70 PRINT R; SPC(15); R%
80 A=4*PI*R↑2
90 V=4*PI*R↑3/3
100 PRINT
110 PRINT "R="; R; "A="; A;
    "V="; V
120 PRINT
130 STOP
140 END
```

Prima linie am scris-o pentru ștergerea ecranului (vezi conversația 2), iar în linia 35 am definit noi valoarea constantei PI (vezi conversația 2). Cum și în BASIC-80 distingem variabile numerice întregi și reale, am procedat la evidențierea prin program a acestei importante facilități introducînd instrucțiunile din liniile 50, 60, 70. Pentru tabularea impusă prin proiectare am utilizat în cadrul instrucțiunilor **PRINT** (liniile 60 și 70) funcția **SPC** care

are aceeaşi semnificaţie ca şi la BASIC-AMSTRAD. Tot atît de bine putem utiliza şi funcţia **TAB** scriind în acest sens instrucţiunile din liniile 60 şi 70 sub forma:

```
60 PRINT "RAZA (Nr. real)"; TAB (21); "RAZA (Nr. întreg)"
70 PRINT R; TAB (21); R%
```

Instrucţiunile **STOP** şi **END** termină execuţia programului redînd controlul sistemului.

Observaţie. În versiunea exitnsă BASIC-80 comentariile se pot pune între ghilimele (exemplu: 10 R=3 'iniţializare').

Instrucţiunea LPRINT

Prezentăm în continuare o variantă simplificată a programului:

```
10 REM "Programul citeşte raza (R) a
unui rezervor sferic",
20 REM "calculează aria (A), volumul (V)
şi tipăreşte:"
30 REM "Raza, Aria, Volumul"
35 PI=3.14159
40 PRINT "RAZA=";
50 INPUT R
60 LET A=4*PI*R↑2
70 LET V=4*PI*R↑3/3
80 LPRINT "R="; R; "A="; A;
"V="; V
90 END
```

în care am urmărit ca afişarea rezultatelor să se facă la imprimanta sistemului. În acest sens am scris instrucţiunea 80:

```
80 LPRINT "R="; R; "A="; A; "V="; V
```

înlocuind pe **PRINT** cu **LPRINT**. **LPRINT** tipăreşte valorile expresiilor din listă la imprimantă. Formatul general este:

Format general
⟨nr. linie⟩**LPRINT**[⟨listă exp.⟩]

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii şi corectării interactive a programelor BASIC-80 sînt:

- linia program poate fi tastată numai după apariţia cursorului;
- pentru introducerea unui cuvînt cheie BASIC în calculator puteţi utiliza indiferent MAJUSCULE sau minuscule;
- pentru ştergerea memoriei înaintea introducerii unui nou program se utilizează comanda **NEW**;
- comanda **CLEAR** iniţializează toate variabilele numerice cu zero;
- comanda **LIST** listează o parte sau tot programul aflat în memorie. Formatul general al comenzii este:

Format general
LIST [⟨nr. linie⟩]

Format general

LIST [⟨nr. linie 1⟩[–[⟨nr. linie 2⟩]]]

Dacă ⟨nr. linie⟩ este omis, programul se listează integral de la linia cu cel mai mic număr. Dacă se indică ⟨nr. linie⟩ se afişează numai linia cu acest număr! Dacă este specificat numai ⟨nr. linie 1⟩ se listează toate liniile program cu numărul mai mare sau egal cu acesta. Dacă se specifică numai ⟨nr. linie 2⟩ se listează toate liniile de la inceputul programului pînă la linia respectivă, inclusiv. În sfîrşit, dacă ambele numere de linii sînt specificate se listează zona de program cuprinsă între ele, inclusiv.

În tabelul 3.5 sînt prezentate cîteva exemple de utilizare a comenzii;

Tabelul 3.5

Comandă	Funcţiune
LIST	listează programul începînd cu linia cu cel mai mic număr
LIST 50	afişează linia 50
LIST –250	listează toate liniile de la începutul programului pînă la linia 250 inclusiv
LIST 150–250	listează toate liniile program cuprinse cu numerele 150 pînă la 250 inclusiv

- comanda **LLIST** listează o parte sau tot programul la imprimantă (cu 132 caractere pe linie); Formatul general este:

Format general

LLIST [⟨nr. linie 1⟩[–[⟨nr. linie 2⟩]]]

- listarea programului se poate întrerupe prin **CTRL/C**;
- pentru ca linia să fie prelucrată se va acţiona tasta **[RETURN]**;
- ştergerea ultimului caracter tastat se realizează prin **CTRL/H**;
- tabularea din 8 în 8 coloane se realizează prin **CTRL/I**;
- **CTRL/R** reafişează linia tastată;
- **CTRL/U** şterge linia curentă;
- pentru tastarea majusculelor şi a simbolurilor superioare înscrise pe taste, menţineţi apăsată tasta **[SHIFT]**;
- **AUTO** generează automat un număr de linie. Formatul general este:

Format general

AUTO[⟨nr. linie⟩[,⟨increment⟩]]

Valorile implicite pentru ⟨nr. linie⟩ şi ⟨increment⟩ sînt 10. Dacă un număr de linie există deja, se va afişa caracterul " * " semnalînd că linia ce urmează va înlocui o linie existentă. În tabelul 3.6 se prezintă cîteva exemple de utilizare a comenzii;

Tabelul 3.6

Comandă	Funcţiune
AUTO	generează numerele de linie: 10, 20, 30, 40, ...
AUTO 18	generează numerele de linie: 18, 28, 38, ...
AUTO 200, 50	generează numerele de linie: 200, 250, 300, ...

Observaţie. Ieşirea din modul „**AUTO**" se realizează prin **CTRL/C**.

● comanda **DELETE** şterge linii din program. Formatul general este:

Format general
DELETE[⟨nr. linie⟩][–⟨nr. linie⟩]

Observaţie. Dacă linia pe care dorim s-o ştergem nu există în program, se afişează un mesaj de eroare.

● comanda **EDIT** permite modificarea unei porţiuni dintr-o linie program fără a mai reintroduce întreaga linie. Formatul general este:

Format general
EDIT⟨nr. linie⟩

Observaţie. Intrarea în mod editare pe linia curentă se poate face şi cu **CTRL/A**. În tabelul 3.7 este prezentat subsetul de comenzi de editare.

Tabelul 3.7

Comandă	Funcţiune
" "	mută cursorul la dreapta.
⟨RUBOUT⟩	mută cursorul la stînga.
I ⟨TEXT⟩ $	inserează text pe poziţia curentă a cursorului.
X	extinde linia; mută cursorul la sfîrşitul liniei unde se poate insera ⟨TEXT⟩ ⟨CR⟩.
nD	şterge n caractere la dreapta cursorului.
H	şterge toate caracterele de la dreapta cursorului.
nS ⟨ch⟩	caută a n-a apariţie a caracterului ⟨ch⟩ şi poziţionează cursorul înainte de el.
nK ⟨ch⟩	şterge a n-a apariţie a caracterului ⟨ch⟩.
C ⟨ch⟩	înlocuieşte următorul caracter cu ⟨ch⟩.
⟨CR⟩	se salvează modificările făcute în linie şi se iese din modul **EDIT**; se tipăreşte restul liniei
E	idem ⟨CR⟩ minus tipărire
Q	se dă controlul interpretorului BASIC-80 după ce se salvează orice modificare făcută cu **EDIT**.
L	listează restul liniei şi poziţionează cursorul la începutul liniei.
A	permite o nouă editare a unei linii.

Salvarea unui program BASIC pe dischetă

Pentru salvarea unui program BASIC-80 pe o dischetă, se foloseşte comanda **SAVE** al cărei format general este:

Format general
SAVE "⟨nume fişier⟩" [, **A** , **P**]

Dacă ⟨nume fişier⟩ există deja pe dischetă, programul salvat va fi scris peste conţinutul vechiului fişier. Parametrul **A** determină salvarea programului în format ASCII, iar parametrul **P** conduce la protejarea programului.

Aplicaţie. Salvaţi programul pe o dischetă (iniţializată!)* sub numele PC380. Verificaţi (vezi comanda **DIR** sub CP/M) dacă programul a fost salvat.

Încărcarea unui program de pe dischetă

Programele de pe dischetă se încarcă în memoria calculatorului cu ajutorul comenzii **LOAD** al cărei format general este:

Format general
LOAD "⟨nume fişier⟩" [, **R**]

LOAD încarcă un fişier de tip ·**BAS** (sub CP/M) în memorie. Comanda **LOAD** închide înaintea încărcării toate fişierele deschise anterior şi şterge din memorie toate variabilele şi liniile program rezidente. **LOAD** cu opţiunea **R** încarcă şi execută programul de pe dischetă.

Observaţie. Opţiunea **R** poate fi folosită şi pentru înlănţuirea programelor ori a segmentelor aparţinînd aceluiaşi program.

Aplicaţie. Încărcaţi în memorie, de pe dischetă, programul PC380.

Pentru adăugarea unui program înregistrat pe dischetă la programul curent din memorie puteţi utiliza instrucţiunea **MERGE** al cărei format general este:

Format general
MERGE "⟨nume fişier⟩"

Dacă o linie din ⟨nume fişier⟩ are acelaşi număr cu o linie din programul rezident în memorie, linia din ⟨nume fişier⟩ înlocuieşte linia din memorie.

Observaţie. După executarea unei instrucţiuni **MERGE,** se intră în modul de lucru direct.

Execuţia programului. Reguli generale

Pentru lansarea în execuţie a unui program BASIC-80, trebuie cunoscute şi respectate următoarele **reguli** generale:

- comanda **RUN** execută un program aflat în memorie. Formatul general este:

Format general
RUN[⟨nr. linie⟩]

Dacă se specifică ⟨nr. linie⟩ atunci se începe execuţia cu acea linie;

- comanda **RUN** încarcă un fişier program de pe dischetă şi îl execută. Formatul general este:

---

* Calculatoarele M 118, JUNIOR şi TPD folosesc dischete de 8" sau 5".

Format general
RUN"⟨nume fişier⟩"[**R**]

RUN închide toate fişierele (deschise) şi şterge conţinutul memoriei înainte de a încărca un program de pe dischetă. Opţiunea **R** permite păstrarea tuturor fişierelor de date deschise anterior;

Observaţie. ⟨nume fişier⟩. reprezintă numele din instrucţiunea **SAVE** cu care s-a salvat programul pe dischetă.

- instrucţiunea **STOP** încheie execuţia programului şi redă controlul interpretorului BASIC; Formatul general este:

Format general
STOP

- instrucţiunea **STOP** nu determină închiderea fişierelor. Ori de cîte ori se întîlneşte un **STOP** în program se va afişa mesajul:

Break in line nnnn

- întreruperea momentană a execuţiei unui program se realizează prin **CTRL/C**;
- **CTRL/O** suspendă o aşteptare de intrare/ieşire (I/E) şi continuă execuţia programului. Un nou **CTRL/O** activează I/E;
- pentru urmărirea modului de execuţie a unui program se utilizează instrucţiunile **TRON** şi **TROFF** ale căror formate generale sînt:

Format general
TRON

Format general
TROFF

TRON determină tipărirea între "[ ]" a numerelor liniilor executate. Instrucţiunea **TROFF** anulează pe **TRON**;

- după introducerea datelor prin instrucţiunea **INPUT** se tastează **[RETURN]**.

Executaţi programul PC380 (de pe dischetă!) cu una din comenzile învăţate. Conversaţia pe care o purtaţi cu calculatorul este cea obişnuită. La apariţia mesajului "RAZA=?" tastaţi valoarea razei pentru care doriţi să calculaţi aria şi volumul rezervorului (de exemplu 3.5), după care reflectaţi puţin la ,,replica" partenerului dvs. de conversaţie. Dacă pentru A şi V vi s-au comunicat valorile 153.938 respectiv 179.594 înseamnă că dialogul dvs. cu cele trei personaje: M 118, JUNIOR şi TPD şi-a atins scopul. Felicitări! În caz contrar, reluaţi conversaţia.

```
R =  3.5   A =  153.938   V =  179.594
R =  1.78   A =  39.8153   V =  23.6237
R =  10.2   A =  1307.4   V =  4445.17
R =  15   A =  2827.43   V =  14137.2
```

Fig. 3.11

Vă prezentăm în fig.3.11 rezultatul execuției programului variantă (vezi pag. 118) obținut la imprimantă.

Ce puteți spune despre forma de editare a celor trei valori? Pare dezordonată, nu-i așa? Lipsesc alinierile! Vom remedia aceste neajunsuri în conversația imediat următoare.

Aplicație. Scrieți un program BASIC care rezolvă următoarele sisteme liniare de două ecuații cu două necunoscute.

a) $\begin{cases} 2x+5y=8 \\ 3x-2y=-7 \end{cases}$ **b)** $\begin{cases} 3x-2y=1 \\ 5x+y=6 \end{cases}$ **c)** $\begin{cases} 12x+5y=0 \\ 8x+10y=8 \end{cases}$

Indicație. Pentru rezolvarea acestor sisteme se utilizează formulele de calcul:

$$x=\frac{b_2k_1-b_1k_2}{a_1b_2-a_2b_1}; \qquad y=\frac{a_1k_2-a_2k_1}{a_1b_2-a_2b_1}$$

unde:

a_1, b_1, k_1, a_2, b_2, k_2 au semnificația de mai jos:

$$\begin{cases} a_1x+b_1y=k_1 \\ a_2x+b_2y=k_2 \end{cases}$$

```
10 print "a1, b1, k1";
20 input a1, b1, k1
30 print "a2, b2, k2";
40 input a2, b2, k2
50 d=a1*b2-a2*b1
60 x=(b2*k1-b1*k2)/d
70 y=(a1*k2-a2*k1)/d
75 print
80 print "X="; x, "Y="; y

run
a1, b1, k1? 2, 5, 8
a2, b2, k2? 3, -2, -7
X=-1     Y=2
```

```
run
a1, b1, k1? 3, -2, 1
a2, b2, k2? 5, 1, 6
X=1      Y=1
run

a1, b1, k1? 12, 5, 0
a2, b2, k2? 8, 10, 8
X=-0.5   Y=1.2
ready
```

d) Să se calculeze valoarea expresiei:

$$E=\frac{a+b-1}{a+b}+\frac{1}{2}\left(\frac{a+b-1}{a+b}\right)^2+$$

$$+\frac{1}{3}\left(\frac{a+b-1}{a+b}\right)^3+\frac{1}{4}\left(\frac{a+b-1}{a+b}\right)^4+$$

$$+\frac{1}{5}\left(\frac{a+b-1}{a+b}\right)^5+\frac{1}{6}\left(\frac{a+b-1}{a+b}\right)^6$$

în care valorile lui a și b se vor introduce dinamic.

```
10 PRINT "A, B";
20 INPUT A, B
30 C=1-1/(A+B)
40 E=C+C↑2/2+C↑3/3+C↑4/4+
   +C↑5/5+C↑6/6
50 PRINT "E="; E
60 END

RUN
A, B? 2,3
E=
```

□ Codificarea în limbajul BASIC-PLUS

vol. 2, pag. 204

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-PLUS trebuie cunoscute și respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- numărul liniei este cuprins între 1 și 32767;
- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- numele unei variabile este format dintr-o literă sau dintr-o literă urmată de o singură cifră;
- variabilele admise sînt de următoarele tipuri: variabile simple numerice întregi, variabile simple numerice reale, variabile simple nenumerice (șir de caractere), variabile indexate numerice (întregi și reale), variabile indexate nenumerice (șir de caractere);
- variabila simplă întreagă este identificată prin orice literă (urmată opțional de o cifră) urmată de %;
- variabila simplă reală este identificată printr-o literă urmată opțional de o cifră;
- variabila simplă șir este identificată printr-o literă (urmată opțional de o cifră) urmată de caracterul $;
- variabila indexată întreagă este identificată prin orice nume de variabilă simplă întreagă;
- variabila indexată reală este identificată prin orice nume de variabilă simplă reală;
- variabila indexată șir este identificată prin orice nume de variabilă simplă șir;
- operatorii aritmetici admiși sînt: "+" (pentru adunare); "−" (pentru scădere); " * " (pentru înmulțire); "/" (pentru împărțire); "**" (pentru ridicare la putere);
- operatorii de relație admiși sînt: "="; "<"; "<="; ">"; ">="; "<>" (diferit); "= =" (aproximativ egal);
- operatorii logici admiși sînt: **NOT, AND, OR, XOR, EQV, IMP**;
- funcțiile matematice admise sînt: **ABS, ATN, COS, EXP, FIX, INT, LOG, LOG 10, NRE** (constantă reală specială e=2.71828), **PI, RAD** (constantă reală specială $\sqrt{2}$=1·41421), **RND, SGN, SIN, SQR**;
- funcțiile nenumerice standard admise sînt: **ASCII, CHR$, DATE$, INSTR, LEFT, LEN, MID, NUM$, RIGHT, SPACES, STRINGS, TIME$, VAL, CVT %$, CVT$ %, CVT$, CVT$F**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, ON...GOTO, IF... ...THEN...ELSE, IF...GOTO**, modificatorul **IF**;
- instrucțiunile de ciclare permise sînt: **FOR...NEXT, FOR WHILE, FOR UNTIL**, modificatorul **WHILE**, modificatorul **UNTIL**, modificatorul **FOR**;
- instrucțiunile de intrare/ieșire permise sînt: **INPUT, INPUT LINE, READ, DATA, RESTORE, PRINT, PRINT USING, MAT READ, MAT INPUT, MAT PRINT**;
- instrucțiunile pentru intrare/ieșire pe disc sînt: **OPEN, CLOSE, INPUT, INPUT LINE, PRINT, PRINT...USING, MAT INPUT, OPEN...AS VIRTUAL, GET, PUT, SAVE, UNSAVE, LOAD, APPEND, LENGTH, REPLACE.**

Vom aplica aceste reguli la scrierea programului.

Ceea ce mai poate reține atenția la acest program este utilizarea lui **TAB** (40) în cadrul instrucțiunilor **PRINT** din liniile 180 și 190 pentru o mai largă spațiere a: capului de tabel (denumirea celei de-a doua coloane se imprimă în coloana 40 și nu în 20 cum eram obișnuiți) și a valorilor lui R, R%, ultima imprimîndu-se începînd tot din coloana 40.

Şi, încă o remarcă foarte importantă pe care trebuie s-o facem! Variabilele numerice R%~0~ (de tip întreg) şi R (de tip real), deşi au acelaşi nume sînt de tipuri diferite (vezi liniile 150, 160, 190).

Observaţie. În BASIC-PLUS comentariile se pot introduce şi prin simbolul "!" (vezi Conversaţia 4).

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii şi corectării interactive a programelor BASIC-PLUS sînt:

- linia program poate fi tastată numai după apariţia prompterului " : ";
- o instrucţiune BASIC-PLUS poate ocupa una sau mai multe linii program, trecerea de pe o linie pe alta indicîndu-se prin caracterul "8" plasat pe ultima poziţie a liniei care se continuă;
- ştergerea unei linii program înainte de a se fi tastat **[RETURN]** se realizează prin acţionarea simultană a tastelor **[CTRL]**, **[U]**;
- în timpul tastării unei linii program anularea unor caractere eronate se realizează prin acţionarea tastei [**DELETE** (RUB OUT)];
- comanda **DELETE** elimină din program un grup de linii. Formatul general este:

Format general
DEL [ETE] ⟨nr. linie⟩ | –⟨nr. linie⟩ | ⟨nr. linie 1⟩–⟨nr. linie 2⟩

Dacă se indică ⟨nr. linie⟩ se elimină toate liniile program cu numărul mai mare sau egal cu acesta. Dacă se specifică – ⟨nr. linie⟩ se elimină toate liniile din program pînă la linia respectivă, inclusiv. Dacă se indică ⟨nr. linie 1⟩ – ⟨nr. linie 2⟩ se elimină zona de program cuprinsă între ele inclusiv. În tabelul 3.8 sînt prezentate cîteva exemple de utilizare a comenzii **DELETE**.

Tabelul 3.8

Comanda	Funcţiune
: **DELETE** 80	elimină toate liniile program cu numărul ⩾80.
: **DEL** –80	elimină toate liniile din program pînă la linia 80, inclusiv.
: **DEL** 200–850	elimină din program toate liniile cuprinse între 200 şi 850 inclusiv.

- comanda **LIST** tipăreşte o parte sau tot programul aflat în memorie. Formatul general al comenzii este:

Format general
LIS[T] [⟨nr. linie⟩ | – ⟨nr. linie⟩ | ⟨nr. linie 1⟩ – ⟨nr. linie 2⟩]

Dacă ⟨nr. linie⟩ este omis, programul se listează integral de la linia cu cel mai mic număr. Dacă se indică ⟨nr. linie⟩, se afişează toate liniile program cu numărul mai mare sau egal cu acesta. Dacă se indică – ⟨nr. linie⟩ se listează toate liniile program pînă la linia respectivă, inclusiv. Dacă se indică ⟨nr. linie 1⟩ – ⟨nr. linie 2⟩ se listează zona de program cuprinsă între ele, inclusiv. În tabelul 3.9 sînt prezentate cîteva exemple de utilizare a comenzii.

Tabelul 3.9

Comanda	Funcţiune
: **LIST** 20	listează toate liniile program cu numărul ⩾20.
: **LIST** –800	listează toate liniile din program pînă la linia 800, inclusiv.

- comanda **NEW** elimină din memoria calculatorului programul rezident. Formatul general este:

Format general

NEW [NO LIST] [⟨nume program⟩]

Prezența parametrului opțional **NO LIST** (elimină informațiile privind textul sursă al programului) împiedică listarea programului. ⟨nume program⟩ reprezintă numele (conține cel mult 9 caractere) sub care se introduce noul program. În tabelul 3.10 se prezintă două exemple în acest sens;

Tabelul 3.10

Comanda	Funcțiune
: **NEW**	elimină din memoria internă programul rezident; ⟨nume program⟩ al programului ce urmează a fi introdus va fi un nume standard BASIC.
: **NEW** M3BPLUS	elimină din memoria internă programul rezident; ⟨nume program⟩ al programului ce urmează a fi introdus va fi: M3BPLUS.

- comanda **RENAME** modifică numele asociat programului utilizator. Formatul general este:

Format general

REN [AME] [NO LIST] [⟨nume program⟩]

Comanda modifică (opțional) și acțiunea parametrului **NOLIST** (v. tabelul 3.11).

Tabelul 3.11

Comanda	Funcțiune
: **RENAME NO LIST** M3BPLUS	asociază programului utilizator numele M3BPLUS și împiedică listarea programului.
: **RENAME**	asociază programului curent numele standard BASIC.

Observație. ⟨nume program⟩ conține cel mult 9 caractere și are aceeași semnificație ca la comanda **NEW**.

Salvarea programelor (din memoria internă) pe un suport extern

Puțină teorie . . . Un program BASIC-PLUS poate fi salvat pe bandă magnetică, disc magnetic etc. Comanda cu care se salvează programul —**SAVE** are formatul general:

Format general

SAV[E] [⟨specificator program⟩]

Așadar, **SAVE** (comandă de bibliotecă!) salvează programul curent utilizator existent în memoria internă pe un suport extern precizat prin

parametrul ⟨specificator program⟩. Formatul general al specificatorului de program este:

Format general
⟨suport⟩ : [⟨grup⟩, ⟨membru⟩] ⟨nume⟩ · ⟨tip⟩ ; ⟨versiune⟩

Pentru moment facem precizarea că ⟨suport⟩ reprezintă unitatea periferică fizică sau logică pe care s-a încărcat fişierul (DK – disk; MM – bandă magnetică; TT – terminal; LP – imprimantă etc.), ⟨tip⟩ precizează natura fişierului: **BAC** – fişier program în format sursă; **DAT** – fişier de date; **TMP** – fişier temporar, iar ⟨nume⟩ reprezintă numele fişierului alcătuit din 9 caractere alfanumerice.

Fişierul creat pe suportul extern primeşte ⟨nume⟩ din ⟨specificator fişier⟩, în caz că acesta a fost precizat, sau numele său intern, în caz contrar. De notat că programul se salvează conform parametrului ⟨tip⟩ indicat.

În tabelul 3.12 se prezintă cîteva exemple de utilizare a comenzii **SAVE**.

Tabelul 3.12

Comandă	Funcţiune
: **SAVE**	salvează programul curent utilizator într-un fişier aflat pe discul sistem, sub numele său intern şi în format sursă.
: **SAVE** DK 2 : P3PLUS.BAC	salvează programul utilizator P3PLUS existent în memoria internă pe discul DK2, în format sursă.
: **SAVE** LP :	listează programul curent utilizator la imprimantă.

Pentru a şterge de pe suportul extern un program existent (salvat cu comanda **SAVE**) se foloseşte comanda **UNSAVE**:

Format general
UNS [**AVE**] ⟨specificator program⟩

Parametrul ⟨specificator program⟩ trebuie să conţină în mod obligatoriu numărul de versiune al fişierului. În caz contrar, comanda nu va fi acceptată.

Aplicaţie. Salvaţi programul existent în memoria internă a calculatorului într-un fişier program M3BPLUS, pe discul sistem, în format sursă.

Încărcarea programelor de pe un suport extern în memoria internă

Pentru a încărca în memoria internă a calculatoarelor INDEPENDENT şi CORAL un program aflat pe un suport extern se utilizează comanda de bibliotecă **LOAD** al cărei format general este:

Format general
LOA[D] [NO LIST] ⟨specificator program⟩

Încărcarea programului se realizează în deplină concordanţă cu informaţiile completate în parametrul ⟨specificator program⟩.

Comanda **LOAD** îndeplineşte următoarele funcţiuni: 1) încarcă programul BASIC de pe suportul extern în memoria internă; 2) înlocuieşte numele programului curent utilizator cu numele indicat în ⟨specificator program⟩; 3) şterge din memorie programul BASIC curent.

De notat că parametrul **[NOLIST]** al comenzii se activează sau nu funcţie atît de comanda respectivă, cît şi de caracteristica programului indicat în ⟨specificator program⟩. Mai precizăm că orice program identificat în ⟨specificator program⟩ cu **BAC** sau orice alt tip (**DAT, TMP**) se consideră program sursă BASIC. În sfîrşit, o sugestie! Înainte de a tasta comanda **LOAD,** salvaţi pe suport extern programul curent utilizator aflat în memoria internă a calculatorului.

Aplicaţie. Încărcaţi de pe discul sistem în memoria internă programul sursă M3BPLUS.

Execuţia programului. Reguli generale

Pentru lansarea în execuţie a unui program BASIC-PLUS trebuie cunoscute şi respectate [51] următoarele **reguli** generale:

- comanda **RUN** al cărei format general este:

Format general
RUN [NO LIST] [⟨specificator program⟩] [, ⟨nr. linie⟩]

lansează în execuţie programul utilizator existent în memoria internă a calculatorului sau pe orice alt suport de memorie externă (bandă, disc etc.), de la prima linie program (în absenţa parametrului ⟨nr. linie⟩) sau de la cea avînd numărul de linie egal cu ⟨nr. linie⟩.
- comanda **RUN** ⟨specificator program⟩ îndeplineşte funcţiunile: 1) elimină din memorie programul utilizator curent (vezi comanda **NEW**); 2) încarcă în memorie programul (fişierul) indicat prin parametrul ⟨specificator program⟩ (vezi comanda **LOAD**); 3) lansează în execuţie programul încărcat;
- la întîlnirea instrucţiunii **STOP,** execuţia programului se opreşte temporar cu mesajul „STOP AT LINE n", n fiind numărul de linie al instrucţiunii;
- la întîlnirea instrucţiunii **END,** execuţia programului este oprită prin mesajul:

END OF PROGRAM AT LINE n

n fiind numărul de linie al instrucţiunii **END**;
- comanda **CONTINUE** al cărei format general este:

Format general
CON[TINUE]

lansează în execuţie programul utilizator (din punctul de întrerupere datorat ultimei instrucţiuni **STOP**);
- comanda **HELP** (vezi Conversaţia 11) explicitează mesajul de eroare ce apare în timpul introducerii sau execuţiei unui program BASIC;
- mesajele de eroare sînt prezentate în vol. 2, pag. 7;
- după introducerea datelor prin instrucţiunea **INPUT** se tastează **[RETURN]**;
- dacă în timpul execuţiei programului se acţionează simultan tastele **[CTRL]** şi **[Z]** programul se opreşte din execuţie cu un mesaj de eroare.

Începeţi conversaţia. Atenţie la ceea ce răspundeţi... prietenului pe care îl aveţi în faţa dvs. (INDEPENDENT sau CORAL) la întrebarea: RAZA=? Vă sugerăm următoarele... posibilităţi: 3; 3.33; 3.0; 3.2 etc. Nu uitaţi ca în finalul conversaţiei, plăcute de altfel, să analizaţi şi dvs. răspunsurile partenerului de conversaţie. În ceea ce ne priveşte (vezi fig. 3.12) vă asigurăm că sînt excelente.

```
:RUN
RAZA = ?3

RAZA ( NR.REAL )                        RAZA ( NR.INTREG )
 3                                       3

R =  3          A =  113.097  V =  113.097

STOP AT LINE 250
:RUN
RAZA = ?3.2

RAZA ( NR.REAL )                        RAZA ( NR.INTREG )
 3.2                                     3

R =  3.2        A =  128.68   V =  137.258

STOP AT LINE 250
```

Fig. 3.12.

☐ Codificarea în limbajul ABASIC

vol. 2, pag. 204

Scrierea programului. Reguli generale.

Pentru scrierea unui program în limbajul ABASIC trebuie cunoscute şi respectate următoarele **reguli** generale:

- Fiecare linie program începe cu un număr de linie;
- numărul liniei este cuprins între 1 şi 32 000;
- o linie program poate să conţină una sau mai multe instrucţiuni separate între ele prin două puncte (" : ");
- o variabilă este identificată printr-un şir de caractere ce începe cu o literă, urmată de un număr oarecare de litere sau cifre;
- variabilele admise sînt de următoarele tipuri: numerice reale, numerice întregi, şiruri de caractere;
- tipul unei variabile se poate stabili şi prin caracterele speciale: "!" sau " # " (pentru variabile numerice reale); "%" (pentru variabile numerice întregi); "$" (pentru şiruri de caractere) ce se ataşează numelui variabilei;
- o variabilă poate fi declarată şi implicit cu ajutorul instrucţiunilor admise: **DEFINT, DEFDBL, DEFSTR**;
- dacă precizarea tipurilor implicite nu este făcută, de utilizator, toate variabilele sînt considerate numerice reale;
- numele unei variabile indexate este orice nume de variabilă acceptat de către ABASIC;
- numărul de dimensiuni ale matricilor declarate este limitat doar de lungimea liniei ABASIC (80 de caractere);
- operatorii aritmetici admişi sînt: "+" (pentru adunare); "–" (pentru scădere); "*" (pentru înmulţire); "/" (pentru împărţire); "**" sau "∧" (pentru ridicarea la putere);

- operatorii de relaţie admişi sînt: "="; "<"; ">"; "<="; ">="; "<>" (diferit) şi "= =" (aproximativ egal);
- operatorii logici admişi sînt: **NOT, AND, OR, XOR,** (SAU exclusiv), **EQV** (echivalenţă) şi **IMP** (implicaţie);
- funcţiile matematice admise sînt: **SIN, COS, TAN, TAG, ATN, PI** (trigonometrice), **SQR, EXP, LOG, LOG 10, LG, INT, ABS, SIGN, SGN, FIX, CINT** (algebrice), **RND, RANDOMIZE** (generarea de nume aleatoare);
- funcţiile admise pentru şiruri de caractere sînt: **LEN, TRM$, POS, INSTR, SEG$, MID$, LEFT$, RIGHT$, STRING$, SRG$, SPACE$, SPC$**;
- funcţiile de conversie admise sînt: **CHR$, CHRA$, VAL, STR$, CHRE$, EBC, ASC, HEX$, OCT$**;
- alte funcţii (auxiliare) disponibile sînt: **TAB, CPOS, BEL$, ERR, ERL, FREE, FRE, EOF, DATE$, DAY$, TIME$**;
- instrucţiunile de salt recunoscute de interpretor sînt: **GOTO, ON GOTO, IF... ...THEN... ELSE, IF...GOTO, IF... THEN... ELSE... ENDIF**;
- instrucţiunile de ciclare permise sînt: **FOR... NEXT, WHILE... ENDWHILE, (WEND)**;
- instrucţiunile de intrare/ieşire permise sînt: **INPUT, LINPUT, (LINE INPUT, INPUT LINE), READ, DATA, RESTORE, PRINT, PRINT USING, WRITE**;
- instrucţiunile admise pentru operaţiile de intrare/ieşire pe disc sînt: **OPEN, CLOSE, INPUT, LINPUT, PRINT, PRINT USING, WRITE, SAVE, LOAD.**

Vom aplica aceste reguli la scrierea programului. Sinceri să fim, mari secrete nu mai avem să vă mărturisim. Comentariile se introduc, de asemenea, cu **REM** sau prin simbolul "!", o variabilă numerică de tip întreg şi una de tip real pot fi definite cu acelaşi nume, **PRINT** cu **INPUT** au semnificaţia ştiută, iar **TAB** (utilizat într-o instrucţiune **PRINT**) setează, de asemenea, poziţia curentă în linia de ieşire, la valoarea dată de expresia din paranteză.

Prezentăm în continuare o variantă simplificată a acestui program, dvs. revenindu-vă sarcina, ca aplicaţie, să adăugaţi şi instrucţiunile care evidenţiază variabilele (inclusiv valori) de tip real şi întreg.

```
10 REM 'PROGRAMUL CITESTE RAZA (R) A UNUI REZERVOR SFERIC'
20 REM 'CALCULEAZA ARIA (A), VOLUMUL (V) SI TIPARESTE:RAZA,ARIA,VOLUMUL'
40 PRINT 'RAZA=';
50 INPUT R
60 LET A=4*PI*R^2
70 LET V=4*PI*R^3/3
80 PRINT 'R = ';R;' A = ';A;' V = ';V
90 END
```

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii şi corectării interactive [36] a programelor ABASIC sînt:

- linia program poate fi tastată numai după apariţia prompterului ">";
- comanda **DELETE** elimină din program un grup de linii (vezi comanda BASIC-PLUS);
- comanda **LIST** tipăreşte o parte sau tot programul aflat în memorie (vezi comanda BASIC-PLUS);
- comanda **NEW** elimină din memoria calculatorului programul rezident (vezi comanda BASIC-80);
- comanda **CLEAR** iniţializează valorile tuturor variabilelor;

- comenzile **RENUM** şi **XCHANGE** permit renumerotarea liniilor program. Formatele generale sînt:

Format general
RENUM ⟨nr. linie început⟩, ⟨pas⟩

Format general
XCHANGE ⟨nr. linie început⟩, ⟨pas⟩

- comanda **EDIT** permite modificarea unei linii. Formatul general este:

Format general
EDIT ⟨nr. linie⟩, ⟨şir de substituit⟩, ⟨şir de intrődus⟩

- comanda **AUTO** are funcţiunea cunoscută;
- trecerea din starea BASIC în starea EDITOR se realizează prin comanda **EXIT**.

Salvarea şi încărcarea programelor

Un program ABASIC poate fi salvat ca fişier ARIEL cu comanda **SAVE**. Formatul general al comenzii este:

Format general
SAVE ⟨nume fişier⟩

Programul poate fi încărcat în memorie, dintr-un alt fişier sau direct de la terminal.

Ştergerea unui program (fişier ARIEL) se realizează cu instrucţiunea **KILL** al cărei format general este:

Format general
KILL ⟨nume fişier⟩

Cu comanda **LOAD** al cărei format general este:

Format general
LOAD ⟨nume fişier⟩

se realizează încărcarea unui fişier ⟨nume fişier⟩ precizat în comanda **SAVE**. Programul curent şi tabela de variabile definite anterior sînt şterse, iar fişierele active sînt închise. Se intră în starea de aşteptare a unei comenzi utilizator. Numerele de linii ale editorului ARIEL devin numere de linii ABASIC.

Observaţie. Dacă la crearea fişierului s-au introdus şi numerele de linii, acestea sînt preferate numerelor de linii ale editorului.

Aplicaţie. Salvaţi programul existent în memoria internă a calculatorului într-un fişier program F3ABAS, pe discul sistem. În pasul imediat următor, încărcaţi programul în memoria internă a calculatorului.

Execuţia programului. Reguli generale

Pentru lansarea în execuţie a unui program ABASIC trebuie cunoscute şi respectate [52] următoarele **reguli** generale:

- comanda **RUN** al cărei format general este:

Format general
RUN [⟨nr. linie⟩]

lansează în execuţie programul de la început sau de la linia indicată ⟨nr. linie⟩;

- comanda **RUN**, cu formatul general:

Format general
RUN [⟨nume fişier⟩ [, ⟨nr. linie⟩]]

încarcă şi lansează în execuţie programul curent sau programul (fişier ARIEL) precizat prin ⟨nume fişier⟩ de la prima linie program (în absenţa parametrului ⟨nr. linie⟩) sau de la cea avînd numărul de linie egal cu ⟨nr. linie⟩;

- comanda **MERGE** permite interclasarea a două fişiere. Formatul general este:

Format general
MERGE ⟨nume fişier⟩ [, ⟨nr. linie⟩]

- la întîlnirea instrucţiunii **STOP** [**END**] execuţia programului se opreşte temporar cu mesajul:

STOPPED AT LINE : n

unde n este numărul de linie al instrucţiunii **STOP.**

- execuţia unui program poate fi oprită la cererea explicită a utilizatorului (se tastează [**BREAK**]) sau în timpul execuţiei unui program (se tastează [**CTRL/C**]);

Şi acum, vă invităm să conversaţi prin intermediul programului F3 ABAS cu calculatorul FELIX-C. Ştiţi şi ce întrebare veţi primi, doar dvs. aţi programat-o! V-aţi pregătit din timp răspunsul? Nu uitaţi că imediat după ce aţi dat "replica" trebuie să tastaţi [**RETURN**].

În sfîrşit, "ultima verba" al calculatorului: STOPPED AT LINE: 90.

Aplicaţie. Se consideră placa plană omogenă din fig. 3.13 avînd dimensiunile date în centimetri. Să se determine coordonatele centrului de greutate.

Indicaţie. Se împarte placa în trei arii elementare şi anume: aria triunghiulară BDC (A_1=1 000 cm^2), aria dreptunghiulară ABDO (A_2=1 200 cm^2) din care s-a decupat semicercul AEO (A_3=–628 cm^2). Alegînd sistemul de axe de coordonate din figură rezultă: X_1=46,66 cm; Y_1=13,3 cm; X_2=15 cm; Y_2=20 cm; X_3=8,5 cm; Y_3=20 cm. Se poate utiliza programul BASIC de la pag. 115, vol. 1. În fig. 3.14 sînt prezentate rezultatele execuţiei programului.

TESTE

a) Scrieţi un program ABASIC care calculează

E=A∗B+C↑D+3

pentru diferite valori de intrare ale lui A, B, C şi D.

```
10 PRINT "A, B, C, D";
20 INPUT A, B, C, D
30 E=A*B+C↑D+3
40 PRINT "E="; E
50 END
RUN
A, B, C, D ? 10, 20, 3, 2, 3
E=212
STOPPED AT LINE : 50
```

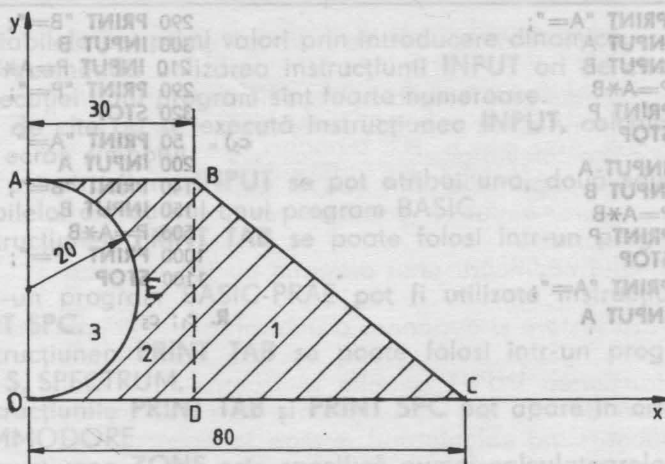


Fig. 3.13.

```
run

A1,A2,A3 ? 1000, 1200,-628

X1,X2,X3 ? 46.66, 15, 8.5

Y1,Y2,Y3 ? 13.3, 20, 20

X = 37.7408821            Y=15.7379135
Ready
```


Fig. 3.14.

b) Scrieți un program ABASIC care, folosind două declarații de tip **INPUT**, are ca rezultat:

b_1) **RUN**
VALOAREA LUI X ? 8
VALOAREA LUI Y ? 6
X+Y=14

b_2) **RUN**
VALOAREA LUI X ? 9
VALOAREA LUI Y ? 3
X–Y=6

b_3) **RUN**
VALOAREA LUI X ? 3
VALOAREA LUI Y ? 7
X*Y=21

b_4) **RUN**
VALOAREA LUI X ? 8
VALOAREA LUI Y ? 2
8/2=4

```
10 PRINT "VALOAREA LUI X";
20 INPUT X
30 PRINT "VALOAREA LUI Y";
40 INPUT Y
50 PRINT "X+Y"; X+Y
60 PRINT "X–Y"; X–Y
70 PRINT "X*Y"; X*Y
80 PRINT "X/Y"; X/Y
90 END
```

c) Care din programele de mai jos (c_1, c_2, c_3, c_4, c_5) calculează corect produsul dintre "A" și "B"?

c_1)
```
10 PRINT "A=";
21 INPUT A
63 PRINT "B=";
69 INPUT B
93 P=A*B
411 PRINT "P="; P
600 STOP
```

c₂)
```
8 PRINT "A=";
33 INPUT A
42 INPUT B
59 P=A*B
50 PRINT P
60 STOP
```
c₃)
```
20 INPUT A
30 INPUT B
25 P=A*B
50 PRINT P
60 STOP
```
c₄)
```
50 PRINT "A=";
200 INPUT A
290 PRINT "B=";
300 INPUT B
210 INPUT P=A*B
290 PRINT "P="; P
320 STOP
```
c₅)
```
50 PRINT "A="
200 INPUT A
140 PRINT "B=";
150 INPUT B
506 P=A*B
1000 PRINT "P="; P
1100 STOP
```
R. c_1; c_5

TEMA 3

□ Răspundeți prin DA sau NU la următoarele întrebări:

– În rezolvarea informatică a unei probleme, întotdeauna se pornește cu datele de ieșire.

– Tabela de variabile conține variabile de intrare, de stare și de ieșire.

– Este important ca schemele și specificațiile de programare să fie verificate împreună cu persoana care a solicitat programul BASIC.

– Diagrama de structură, pseudocodul și schema logică reprezintă principalele instrumente cu care se proiectează un program.

– Pseudocodul trebuie să fie precis spre a putea fi înțeles de programatori.

– Schema logică reprezintă o metodă grafică de documentare a programului.

– Instrucțiunea **REM** poate apare oriunde în cadrul unui program BASIC.

– În BASIC-aMIC se recomandă ca toate comentariile să se introducă între apostrofuri.

– Pe o linie multiinstrucțiune, instrucțiunea **REM** trebuie să fie ultima instrucțiune a liniei.

– În BASIC-PRAE, în locul lui **REM** se poate utiliza și apostroful.

– În BASIC HC-85, TIM S, SPECTRUM în locul lui **REM** se poate utiliza și apostroful.

– În BASIC-AMSTRAD, **REM** poate fi înlocuit printr-un apostrof.

– În BASIC-COMMODORE, **REM** nu poate fi înlocuit printr-un apostrof.

– În BASIC-80, comentariile pot fi precedate de un apostrof.

– În versiunea extinsă BASIC-80, comentariile se pun între ghilimele.

– În BASIC-PLUS, comentariile se introduc prin **REM** sau prin simbolul "!".

– În ABASIC, comentariile se introduc prin **REM** sau prin simbolul "!".

– Variabilele pot primi valori prin atribuire propriu-zisă.

– Variabilele pot primi valori prin introducere dinamică.

– Se recomandă utilizarea instrucţiunii **INPUT** ori de cîte ori datele necesare execuţiei unui program sînt foarte numeroase.

– Ori de cîte ori se execută instrucţiunea **INPUT,** calculatorul aMIC afişează pe ecran simbolul: " : ".

– Cu o instrucţiune **INPUT** se pot atribui una, două sau mai multe valori variabilelor din cadrul unui program BASIC.

– Instrucţiunea **PRINT TAB** se poate folosi într-un program BASIC-aMIC.

– Într-un program BASIC-PRAE pot fi utilizate instrucţiunile **PRINT TAB** şi **PRINT SPC.**

– Instrucţiunea **PRINT TAB** se poate folosi într-un program BASIC HC-85, TIM S, SPECTRUM.

– Instrucţiunile **PRINT TAB** şi **PRINT SPC** pot apare în orice program BASIC-COMMODORE.

– Instrucţiunea **ZONE** este specifică numai calculatoarelor personale AMSTRAD.

– Instrucţiunile **PRINT TAB** şi **PRINT SPC** nu pot apare în programele BASIC-AMSTRAD.

– Instrucţiunile **PRINT TAB** şi **PRINT SPC** pot apare în orice program BASIC-80.

– Instrucţiunea **PRINT TAB** poate fi folosită în orice program BASIC-PLUS.

– Instrucţiunea **PRINT SPC** poate fi folosită în orice program BASIC-PLUS.

– Instrucţiunile **PRINT TAB** şi **PRINT SPC** pot apare în orice program ABASIC.

– Programele BASIC-aMIC pot fi salvate numai pe dischetă.

– În cadrul comenzii **SAVE** de salvare a unui program BASIC-aMIC pe casetă, numele programului este constituit din patru caractere hexazecimale.

– În cadrul comenzii **ASAVE** de salvare a unui program BASIC-PRAE pe casetă, numele programului este constituit din oricîte caractere cuprinse între ghilimele.

– Programele BASIC HC-85, TIM S, SPECTRUM pot fi salvate numai pe dischetă.

– Programele BASIC-AMSTRAD pot fi salvate pe casetă şi pe dischete de 3".

– Programele BASIC-COMMODORE pot fi salvate pe casetă şi pe dischete de 5".

☐ **Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:**

a) Procesul de alcătuire a unui program constă din următoarele faze: analiză, ..., ...

b) În faza de analiză se realizează tabela de variabile şi ....

c) În faza de proiectare se realizează ....

d) Dintre virtuţile diagramei de structură reţinem: .... , ... , ...

e) Pseudocodul se pretează la documentaţia programării ....

f) Acţiunile primitive sînt acţiunile care, spre a putea fi înţelese şi executate de către calculator ....

g) Acţiunile neprimitive sînt opusul acţiunilor ....

h) Lista tuturor acţiunilor primitive constituie .... problemei de rezolvat.

i) În timpul codificării unui program au loc trei paşi: .... , ... , ...

j) În faza testare şi depanare a unui program se constată ....

k) Instrucţiunea **INPUT** permite utilizatorului să aloce diferite valori variabilelor, ori de cîte ori un program este executat, fără ca programul să fie modificat. Cînd calculatorul ajunge la instrucţiunea **INPUT** din cadrul unui program, afişează semnul întrebării (de regulă) şi aşteaptă ca utilizatorul să introducă valoarea ce trebuie alocată variabilei. Utilizatorul trebuie să introducă o valoare (după semnul întrebării) şi apoi să acţioneze tasta [RETURN].

În fig. 3.15 se prezintă lista unui program BASIC şi rezultatul execuţiei acestuia.

```
10  INPUT B
20  PRINT "B="; B
30  END
RUN
? 5
B = 5 ←—— Valoarea lui B este
          afişată după semnul
          egal
```

Fig. 3.15.

După ce am tastat **RUN** şi am acţionat tasta [RETURN], calculatorul a afişat .... După aceea, am tastat .... , ceea ce reprezintă valoarea variabilei de intrare .... Calculatorul afişează apoi şirul de caractere "B=" urmat de .... lui B.

Programul poate fi din nou executat, dînd o valoare diferită variabilei B. Precizaţi modul de conversaţie cu calculatoarele: aMIC, PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE M-118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX-C dacă utilizatorul introduce pentru B valoarea 20?

RUN

....

....

l) Adăugaţi la programul precedent instrucţiunea:

5 **PRINT** "B=";

Ați remarcat punctul și virgula de la sfîrșitul instrucțiunii?

Cînd se folosește ";" la sfîrșitul unei instrucțiuni **PRINT** ....

În fig. 3.16(a) se prezintă noul program și începutul execuției acestuia. Acum știm sigur ce dorește calculatorul – o valoare pentru variabila de intrare B. Vom atribui lui B valoarea 308, o vom scrie după semnul întrebării și vom acționa tasta **[RETURN]** (v. fig. 3.16(b)).

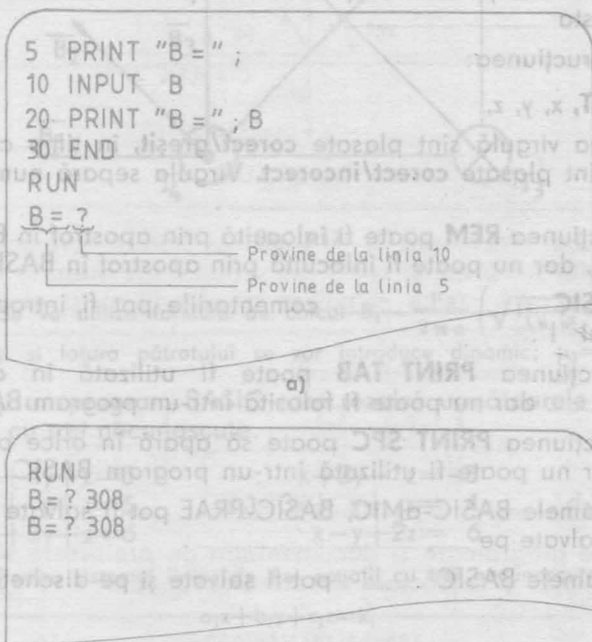


Fig. 3.16 a–b.

Precizați eroarea din fig. 3.16(b) cît și modul de conversație cu toată gama de calculatoare studiate, dacă utilizatorul introduce pentru B valoarea 902.

RUN ....

m) Cînd un program care conține o instrucțiune **INPUT** cu mai multe variabile este executat pe calculatoarele .... prima valoare afișată de utilizator (după semnul întrebării) este destinată .... variabile ce apare în instrucțiunea **INPUT**; a .... valoare tipărită de utilizator va fi destinată celei de-a doua variabile din instrucțiunea **INPUT** etc. Cele două variabile din programul care conține instrucțiunea **INPUT**, precum și valorile introduse de utilizator în timpul execuției programului, trebuie să fie separate de ....

n) Se consideră următorul program BASIC:

```
10 PRINT "VALORILE LUI X ŞI Y"
20 INPUT X, Y
30 PRINT "X="; X; Y="; Y
40 END
```

Dorim să atribuim lui X și Y valorile 3, respectiv 8. Pentru aceasta, tastăm .... după care ....

o) Ori de cîte ori se execută instrucțiunea **INPUT**, calculatoarele ...., ..., ... afișează pe ecran semnul întrebării.

p) În momentul execuției unei instrucțiuni **INPUT**, calculatorul așteaptă (răbdător!) pînă tastați valoarea pe care doriți să o introduceți și pînă cînd veți acționa tasta ....

r) În instrucțiunea:

30 **INPUT**, x, y, z,

prima și ultima virgulă sînt plasate **corect/greșit**, în timp ce a doua și a treia virgulă sînt plasate **corect/incorect**. Virgula separă numai ....

s) Instrucțiunea **REM** poate fi înlocuită prin apostrof în BASIC ...., dar nu poate fi înlocuită prin apostrof în BASIC ....

t) În **BASIC** .... comentariile pot fi introduse prin **REM** și prin simbolul "!".

u) Instrucțiunea **PRINT TAB** poate fi utilizată în orice program BASIC .... dar nu poate fi folosită într-un program BASIC ....

v) Instrucțiunea **PRINT SPC** poate să apară în orice program BASIC .... dar nu poate fi utilizată într-un program BASIC ....

w) Programele BASIC-aMIC, BASIC-PRAE pot fi salvate pe .... dar nu pot fi salvate pe ....

x) Programele BASIC .... pot fi salvate și pe dischetă și pe casetă.

□ Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Să se calculeze momentele de inerție axiale (IZ=IY) ale unei secțiuni circulare de rază R.

Indicație. Se utilizează formula $Iy=Iz=\frac{\pi d^4}{64}$, unde d este diametrul secțiunii. Valoarea lui d se va introduce dinamic.

b) Să se calculeze momentele de inerție axiale ale unei suprafețe inelare avînd diametrul exterior D și diametrul interior d.

Indicație. Se va utiliza formula de calcul $Iz=Iy=\frac{\pi}{64}(D^4-d^4)$. Valorile celor două diametre se vor introduce dinamic (exemplu: D=40 cm; d=20 cm).

c) Prin vîrfurile unui pătrat cu latura a=10 cm trec patru conductoare rectilinii, foarte lungi, perpendiculare pe planul figurii 3.17, parcurse de curent în sensul indicat pe figură. Dacă intensitățile curenților au valorile $I_1=1A$, $I_2=2A$, $I_3=1A$ să se determine inducția magnetică B_4 în vîrful patru al pătratului.

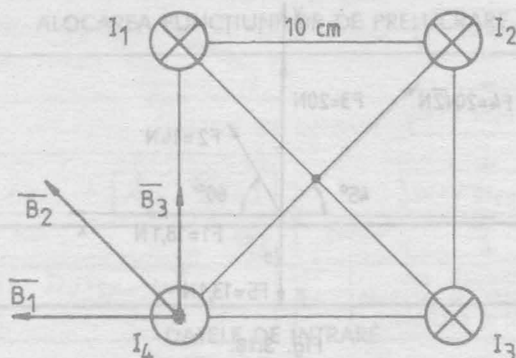


Fig. 3.17.

Indicație. Se va utiliza formula de calcul $B_4 = \frac{\mu_0}{2\pi a}\left(\sqrt{I_1^2 + I_3^2} + I_2/\sqrt{2}\right)$. Valorile celor trei curenți și latura pătratului se vor introduce dinamic; $\mu_0 = 4\pi \cdot 10^{-7}$ .N/A².

d) Scrieți un program BASIC care rezolvă următoarele sisteme liniare de trei ecuații cu trei necunoscute.

$$d_1)\ \begin{cases} 2x+4y+z=5 \\ x+y+z=6 \\ 2x+3y+z=6 \end{cases} \qquad d_2)\ \begin{cases} x+2y-z=-3 \\ 3x+y+z=4 \\ x-y+2z=6 \end{cases} \qquad d_3)\ \begin{cases} x-y=5 \\ y-z=-6 \\ 2x-z=2 \end{cases}$$

Indicație. Pentru sistemul liniar de trei ecuații cu trei necunoscute

$$a_1x+b_1y+c_1z=k_1$$
$$a_2x+b_2y+c_2z=k_2$$
$$a_3x+b_3y+c_3z=k_3$$

x, y și z (necunoscutele) au expresiile:

$$x = \frac{b_2c_3k_1+b_1c_2k_3+b_3c_1k_2-b_2c_1k_3-b_3c_2k_1-b_1c_3k_2}{\Delta_3}$$

$$y = \frac{a_1c_3k_2+a_3c_2k_1+a_2c_1k_3-a_3c_1k_2-a_1c_2k_3-a_2c_3k_1}{\Delta_3}$$

$$z = \frac{a_1b_2k_3+a_3b_1k_2+a_2b_3k_1-a_3b_2k_1-a_1b_3k_2-a_2b_1k_3}{\Delta_3}$$

unde:

$$\Delta_3 = a_1b_2c_3+a_3b_1c_2+a_2b_3c_1-a_3b_2c_1-a_1b_3c_2-a_2b_1c_3$$

e) Asupra unui punct material acționează un sistem de forțe alcătuit din cinci forțe concurente (vezi fig. 3.18). Să se determine rezultanta acestui sistem de forțe.

Indicație. Se va utiliza relația de calcul $\ddot{R} = \sqrt{Rx^2+Ry^2}$, unde $Rx = F_1x+F_2x+\ldots, F_5x$ iar $Ry = F_1y+F_2y+\ldots, F_5y$. Rx și Ry reprezintă suma algebrică a proiecțiilor forțelor pe axele Ox respectiv Oy. Valorile lui $F_1x, \ldots, F_5x$ și ale lui $F_1y, \ldots, F_5y$ se vor introduce dinamic.

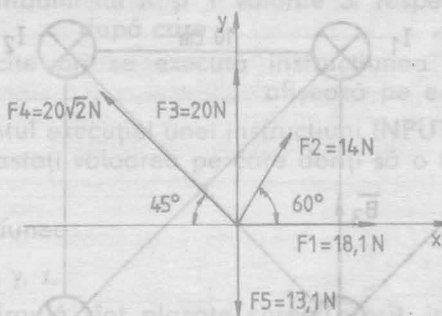


Fig. 3.18.

f) Un mobil efectuează o mişcare oscilatorie armonică. Ştiind că pentru elongaţiile $x_1=2$ cm şi $x_2=3$ cm, mobilul are vitezele $v_1=5\ m\cdot s^{-1}$ şi respectiv $v_2=4\ m\cdot s^{-1}$, să se calculeze amplitudinea şi perioada mişcării oscilatorii a mobilului.

Indicaţie. Se vor utiliza formulele de calcul:

$$A=\sqrt{\frac{x_1^2v_2^2-x_2^2v_1^2}{v_2^2-v_1^2}};\quad T=2\pi\sqrt{\frac{x_1^2-x_2^2}{v_2^2-v_1^2}}$$

Valorile vitezelor şi elongaţiilor se vor introduce dinamic.

Specificaţiile de programare şi documentaţia de proiectare sînt ilustrate în modulele de analiză şi proiectare structurată, fig. 3.19.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieşire
V1 – viteza corespunzătoare elongaţiei X1	C : diferenţa pătratelor vitezelor	A : amplitudinea mişcării oscilatorii
V2 – viteza corespunzătoare elongaţiei X2		T : perioada mişcării oscilatorii
X1 : prima elongaţie		
X2 : a doua elongaţie		

a

SPECIFICAŢII DE PROGRAMARE

Descrierea programului
Programul citeşte de la terminal elongaţiile mişcării oscilatorii şi vitezele mobilului, calculează amplitudinea şi perioada mişcării oscilatorii şi afişează rezultatul.

Intrări
Două elongaţii şi două viteze.

Ieşiri
Perioada şi amplitudinea

Lista de funcţiuni ale programului
1. Citeşte date intrare
2. Calculează amplitudinea mişcării oscilatorii
3. Calculează perioada mişcării oscilatorii
4. Afişează rezultatul
5. Stop.

b

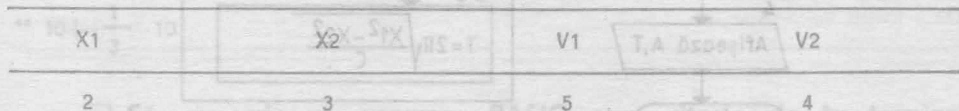
Fig. 3.19. a–g. Modulele de analiză şi proiectare structurată: *a*) tabela de variabile; *b*) specificaţii de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Modul	Funcțiuni
CIT–DAT	1
PREL–DAT	2,3
LIST	4
SFIRȘIT	5

c)

DATELE DE INTRARE



X1	X2	V1	V2
2	3	5	4

d)

APLICAȚIE



e)

PSEUDOCOCUL

```
APLICAȚIE     SEQ
CIT–DAT       SEQ
                INPUT X1, X2
                INPUT V1, V2
CIT–DAT       END
PREL–DAT      SEQ
                C=V2 * V2–V1 * V1
                A=((X1 ↑ 2 * V2 ↑ 2–X2 ↑ 2 * V1 ↑ 2)/C) ↑ ·5
                T=2 * PI * ((X1 ↑ 2–X2 ↑ 2)/C) ↑ ·5
PREL–DAT      END
                PRINT "A="; A, "T="; T
APLICAȚIE     END
```

f)

Fig. 3.19 c) alocarea funcțiunilor de prelucrare; *d*) datele de intrare; e) diagrama de structură; *f*) pseudocodul;

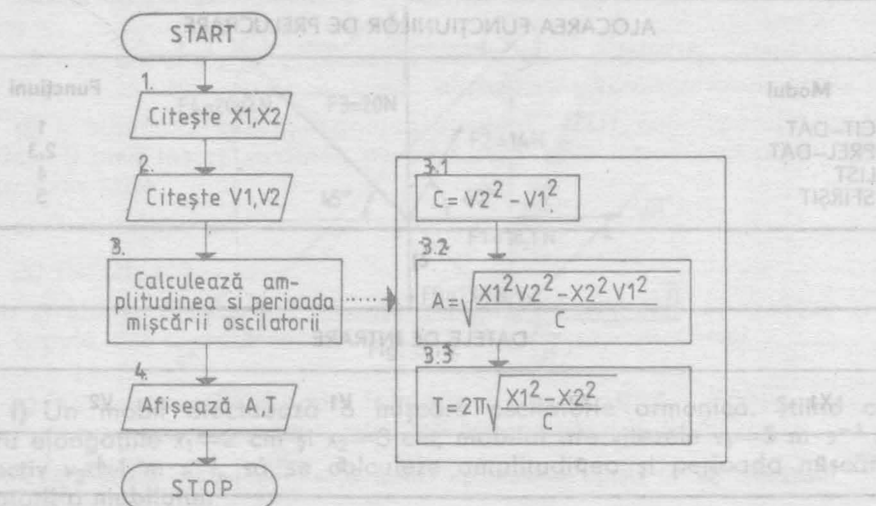


Fig. 3.19

g) schema logică.

g) Se consideră placa plană omogenă din figura 3.20, avînd dimensiunile date în centimetri. Să se determine coordonatele centrului de greutate.

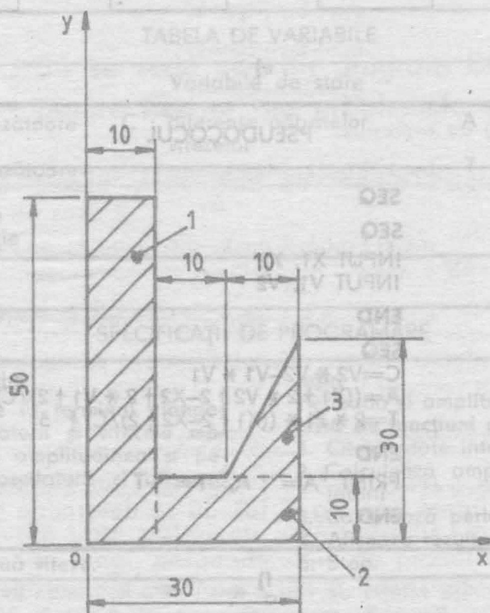


Fig. 3.20.

Indicaţie. Se împarte placa în trei arii elementare şi se alege sistemul de axe din figură.

Datele de intrare sînt prezentate în tabelul 3.13.

Tabelul 3.13

DATELE DE INTRARE

A1	A2	A3	X1	X2	X3	Y1	Y2	Y3
500	200	100	5	20	26,666*	25	5	13,333**

* $20+\frac{2}{3}\cdot 10$

** $10+\frac{1}{3}\cdot 10$

☐ **Să se scrie un program BASIC care citeşte de la terminal** cantităţile lunare de jucării (pentru un articol) fabricate de întreprinderea "URSULEŢUL", calculează cantităţile totale trimestriale şi procentele corespunzătoare din cantitatea totală anuală.

Specificaţiile de programare şi documentaţia de proiectare sînt ilustrate în modulele de analiză şi proiectare structurată, fig. 3.21.



a)

DATELE DE INTRARE*

A1	A2	A3	B1	B2	B3	C1	C2	C3	D1	D2	D3
10	20	30	5	6	7	15	20	25	30	5	8

* mii bucăţi

b)

Fig. 3.21 a–h. Modulele de analiză şi proiectare structurată: *a*) formatul datelor de ieşire; *b*) datele de intrare;

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
A1 : livrări ianuarie	T1 : livrări trim. I	P1 : procent trim. I
A2 : livrări februarie	T2 : livrări trim. II	P2 : procent trim. II
A3 : livrări martie	T3 : livrări trim. III	P3 : procent trim. III
B1 : livrări aprilie	T4 : livrări trim. IV	P4 : procent trim. IV
B2 : livrări mai	T : total livrări	
B3 : livrări iunie		
C1 : livrări iulie		
C2 : livrări august		
C3 : livrări septembrie		
D1 : livrări octombrie		
D2 : livrări noiembrie		
D3 : livrări decembrie		

c)

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal cantitățile lunare de jucării fabricate, calculează cantitățile totale trimestriale și procentele corespunzătoare din cantitatea totală anuală și afișează rezultatele.

Intrări

Cantitățile lunare de jucării fabricate.

Ieșiri

Cantitățile trimestriale de jucării fabricate, exprimate în procente.

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează cantitățile totale trimestriale.
3. Calculează cantitatea totală anuală exprimată în procente
4. Calculează cantitățile trimestriale de jucării fabricate
5. Afișează rezultatul
6. Stop

d)

ALOCAREA FUNCȚIUNILR DE PRELUCRARE

Modul	Funcțiuni
CIT–DAT	1
PREL–DAT	2, 3, 4
LIST	5
SFIRȘIT	6

e)

c) tabela de variabile; d) specificații de programare; e) alocarea funcțiunilor de prelucrare;

PSEUDOCODUL

```
TEMA3-4      SEQ
               INPUT A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3
PREL-DAT     SEQ
               T1=A1+A2+A3
               T2=B1+B2+B3
               T3=C1+C2+C3
               T4=D1+D2+D3
               T =T1+T2+T3+T4
               P1=(T1 * 100)/T
               P2=(T2 * 100)/T
               P3=(T3 * 100)/T
               P4=(T4 * 100)/T
PREL-DAT     END
               PRINT P1, P2, P3, P4
TEMA3-4      END
```

f)

TEMA

CIT-DAT | PREL-DAT | LIST | SFIRSIT

g)

START

Citește A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3

T1 = A1 + A2 + A3
T2 = B1 + B2 + B3
T3 = C1 + C2 + C3
T4 = D1 + D2 + D3
T = T1 + T2 + T3 + T4

P1 = (T1 * 100) / T
P2 = (T2 * 100) / T
P3 = (T3 * 100) / T
P4 = (T4 * 100) / T

Afișează P1, P2, P3, P4

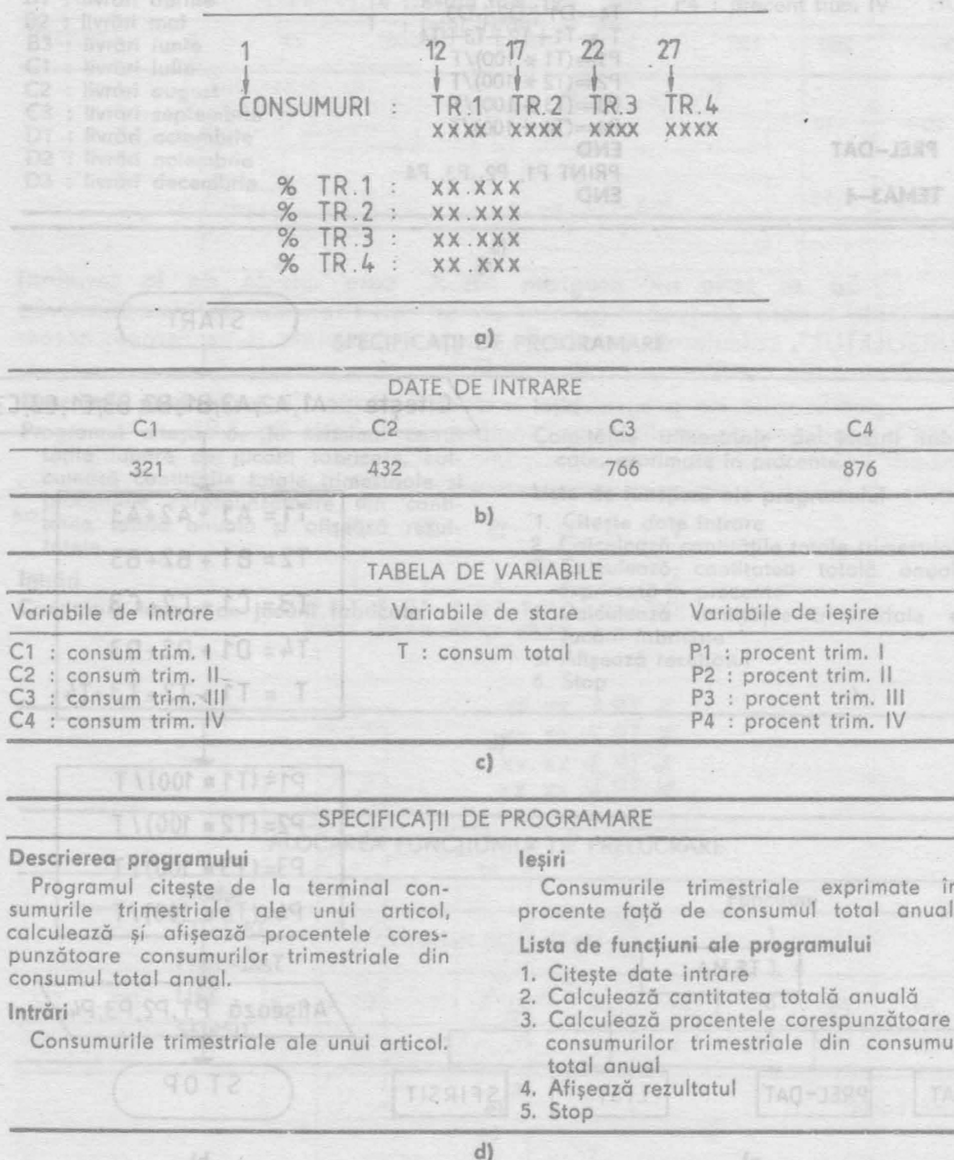
STOP

h)

f) pseudocodul; *g)* diagrama de structură; *h)* schema logică.

□ **Să se scrie un program BASIC care citește de la terminal** consumurile trimestriale ale unui articol, calculează și afișează procentele corespunzătoare consumurilor trimestriale din consumul total anual (vezi și L. Dumitrașcu, Învățăm COBOL... conversînd cu calculatorul, Editura Tehnică, București, 1985).

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată, fig. 3.22.



```
1                 12    17    22    27
↓                 ↓     ↓     ↓     ↓
CONSUMURI  :      TR.1  TR.2  TR.3  TR.4
                  XXXX  XXXX  XXXX  XXXX

      % TR.1 :   XX.XXX
      % TR.2 :   XX.XXX
      % TR.3 :   XX.XXX
      % TR.4 :   XX.XXX
```

a)

DATE DE INTRARE

C1	C2	C3	C4
321	432	766	876

b)

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
C1 : consum trim. I	T : consum total	P1 : procent trim. I
C2 : consum trim. II		P2 : procent trim. II
C3 : consum trim. III		P3 : procent trim. III
C4 : consum trim. IV		P4 : procent trim. IV

c)

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal consumurile trimestriale ale unui articol, calculează și afișează procentele corespunzătoare consumurilor trimestriale din consumul total anual.

Intrări

Consumurile trimestriale ale unui articol.

Ieșiri

Consumurile trimestriale exprimate în procente față de consumul total anual.

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează cantitatea totală anuală
3. Calculează procentele corespunzătoare consumurilor trimestriale din consumul total anual
4. Afișează rezultatul
5. Stop

d)

Fig. 3.22 a–h. Modulele de analiză și proiectare structurată: *a)* formatul datelor de ieșire; *b)* datele de intrare; *c)* tabela de variabile; *d)* specificații de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Modul	Funcțiuni
CIT–DAT	1
PREL–DAT	2,3
LIST	4
SFIRȘIT	5

e)

PSEUDOCODUL

```
TEMA3–5     SEQ
            INPUT C1, C2, C3, C4
PREL–DAT    SEQ
            T=C1+C2+C3+C4
            P1=(C1 * 100)/T
            P2=(C2 * 100)/T
            P3=(C3 * 100)/T
            P4=(C4 * 100)/T
PREL–DAT    END
            PRINT P1, P2, P3, P4
TEMA3–5     END
```

f)

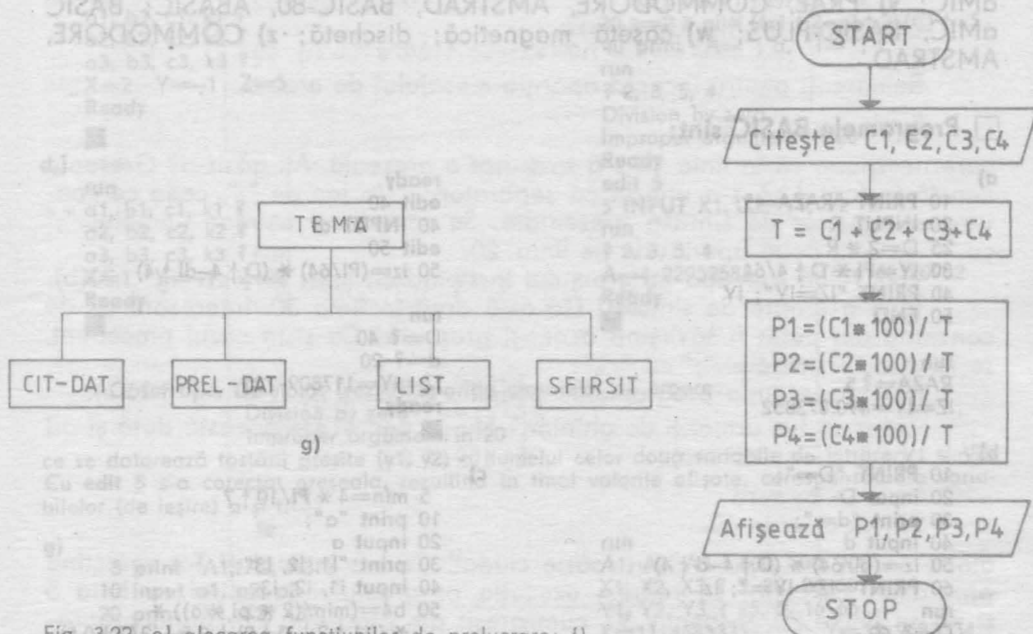


Fig. 3.22. *e)* alocarea funcțiunilor de prelucrare; *f)* pseudocodul; *g)* diagrama de structură; *h)* schema logică.

SOLUȚIA TEMEI 3

☐ **Răspunsurile sînt:**

DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, NU, DA, DA, DA, DA, DA, DA, DA, DA, NU, DA, DA, NU, DA, DA, DA, DA, NU, DA, DA, NU, DA, NU, DA, DA, NU, DA, DA.

☐ **Cuvintele lipsă sînt:**

a) analiză, proiectare, codificare, testare și depanare; **b)** specificațiile de programare; **c)** diagrama de structură, pseudocodul, schema logică; **d)** este independentă de limbaj sau calculator, poate reda o structură a datelor sau a logicii de prelucrare, scoate în evidență structura funcțională a programului, evidențiază necesarul de operații de intrare/ieșire; **e)** structurate; **f)** nu mai necesită informații suplimentare; **g)** primitive; **h)** algoritmul; **i)** scrierea programului (pasul 1), introducerea programului (pasul 2), execuția programului (pasul 3); **j)** dacă programul realizat este în concordanță cu specificațiile de programare; **k)** ?, 5, B, valoarea; **l)** valoarea pe care o introduceți apare pe aceeași linie cu semnul întrebării; **m)** PRAE, HC-85, TIM S, SPECTRUM, M 118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX C, primei variabile, a doua valoare, virgulă; **n)** **RUN, [RETURN]**, tastăm 3, virgulă, 8, **[RETURN]**; **o)** vezi m; **p)** **[RETURN]**; **r)** greșit, corect, variabilele; **s)** PRAE, AMSTRAD, BASIC-80; HC-85, TIM S, SPECTRUM, COMMODORE, BASIC-PLUS, ABASIC; **t)** BASIC-PLUS, ABASIC; **u)** PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS, ABASIC; aMIC; **v)** PRAE, COMMODORE, AMSTRAD, BASIC-80, ABASIC; BASIC aMIC, BASIC-PLUS; **w)** casetă magnetică; dischetă; **z)** COMMODORE, AMSTRAD.

☐ **Programele BASIC sînt:**

a)

```
10 PRINT "RAZA=";
20 INPUT R
25 D=2 * R
30 IY=PI * D ↑ 4/64
40 PRINT "IZ=IY"; IY
50 END

run
RAZA=? 5
IZ=IY=490.873852
```

b)

```
10 PRINT "D=";
20 input D
30 print "d=";
40 input d
50 iz=(pi/64) * (D ↑ 4–d ↑ 4)
60 PRINT "IZ=IY="; iz
run
D=? 40
d=? 20
IZ=IY=0
ready
edit 40
40 INPUT dl
edit 50
50 iz=(PI/64) * (D ↑ 4–dl ↑ 4)

run
D=? 40
d=? 20
IZ=IY=117809·725
ready
■
```

c)

```
 5 min=4 * PI/10 ↑ 7
10 print "a";
20 input a
30 print "I1, I2, I3";
40 input i1, i2, i3
50 b4=(min/(2 * pi * a)) *
   * ((i1 ↑ 2+i3 ↑ 2) ↑ 0.5+i2/2 ↑ 0.5)
55 print
60 print "B4="; b4
```

```
run
a ? · 1
l1, l2, l3 ? 1, 2, 1
B4=5·65685E–06
Ready
■
```

d)

```
10 print "a1, b1, c1, k1";
20 input a1, b1, c1, k1
30 print "a2, b2, c2, k2";
40 input a2, b2, c2, k2
50 print "a3, b3, c3, k3";
60 input a3, b3, c3, k3
70 d=a1*b2*c3+
a3*b1*c2+a2*b3*c1–
a3*b2*c 1–a1*b3*c2–a2*b1*c3
80 d1=b2*c3*k1+b1*c2*k3+
b3*c1*k2–b1*c1*k 3–b3*c2*k1–
b1*c3*k2
90 d2=a1*c3*k1+a3*c2*k1+
a2*c1*k3–a3*c1*k 2–a1*c2*k3–
a2*c3*k1
100 d3=a1*b2*k3+a3*b1*k2+
a2*b3*k1–a3*b2*k1–a1*b3*k2–
a2*b1*k3
110 x=d1/d: y=d2/d: z=d3/d
115 print
120 print "X="; x; "Y="; y; "Z=";
z
```

d₁)

```
run
a1, b1, c1, k1 ?
a2, b2, c2, k2 ?
a3, b3, c3, k3 ?
X=2  Y=–1  Z=5
Ready
■
```

d₂)

```
run
a1, b1, c1, k1 ?
a2, b2, c2, k2 ?
a3, b3, c3, k3 ?
X=1  Y=–1  Z=2
Ready
■
```

d₃)

```
run
a1, b1, c1, k1 ?
a2, b2, c2, k2 ?
a3, b3, c3, k3 ?
X=3  Y=–2  Z=4
Ready
■
```

e)

```
10 print "F1x, F2x, F3x, F4x, F5x";
20 input f1x, f2x, f3x, f4x, f5x
30 print "F1Y, F2Y, F3Y, F4Y, F5Y";
40 input f1y, f2y, f3y, f4y, f5y
50 rx=f1x+f2x+f3x+f4x+f5x
60 ry=f1y+f2y+f3y+f4y+f5y
70 r=(rx↑2+ry)↑2)↑·5
75 print
80 print "R="; r; "N"
90 end
run
F1x, F2x, F3x, F4x, F5x ?
   18·1, 7, 0, –20, 0
F1Y, F2Y, F3Y, F4Y, F5Y ?
   0,12.11,20, 20, –13·1
R=39.3419636 N
Ready
■
```

f)

```
5 input x1, x2, y1, y2
10 c=V2 ↑ 2–V1 ↑ 2
20 a=((x1 ↑ 2 * V2 ↑ 2–
x2 ↑ 2 * V1 ↑ 2)/C) ↑ ·5
30 t=2 * pi * ((x1 ↑ 2–x2 ↑ 2)/C) ↑ ·5
40 print "A="; a, "T="; t
run
? 2, 3, 5, 4
Division by zero
Improper argument in 20
Ready
edit 5
5 INPUT X1, X2, V1, V2
run
? 2, 3, 5, 4
A=4·22952585        T=4·68320982
Ready
■
```

Observaţie. De notat (vezi f) apariţia mesajelor de eroare:

Division by zero
Improper argument in 20

ce se datorează tastării greşite (y1, y2) a numelui celor două variabile de intrare V1 şi V2. Cu **edit** 5 s-a corectat greşeala, rezultînd în final valorile afişate, corespunzătoare variabilelor (de ieşire) a şi t.

g)

```
5 print "A1, A2, A3";
10 input a1, a2, a3
20 print "X1, X2, X3";
30 input x1, x2, x3
40 print "Y1, Y2, Y3";
50 input y1, y2, y3
```

```
run
A1, A2, A3 ? 500, 200, 100
X1, X2, X3 ? 5, 20, 26·60
Y1, Y2, Y3 ? 25, 5, 16·66
X=11·4583333        Y=18·9583334
Ready
■
```

```
60 a=a1+a2+a3
70 x=(a1 * x1+a2 * x2+a3 * x3)/a
80 y=(a1 * y1+a2 * y2+a3 * y3)/a
85 print : print
90 print "X="; x, "Y="; y
100 end
```

☐ **Programul pentru calculul procentelor corespunzătoare** cantităților trimestriale de jucării, fabricate de întreprinderea Ursulețul este:

```
5 print "a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3"
10 input a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3
20 t1=a1+a2+a3; t2=b1+b2+b3; t3=c1+c2+c3
30 t4=d1+d2+d3; t=t1+t2+t3+t4
40 p1=t1 * 100/t; p2=t2 * 100/t; p3=t3 * 100/t
50 p4=t4 * 100/100
60 cls
70 print "JUCĂRII:"; tab (10); "TR. 1"; tab (15); "TR. 2"; tab (20);"
   "TR. 3"; tab (25); "TR. 4"
80 print tab (10); t1; tab (15); t2; tab (20); t3; tab (25); t4
85 print: print
90 print tab (2); "% TR. 1:"; p1
100 print tab (2); "% TR. 2:"; p2
110 print tab (2); "% TR. 3:"; p3
120 print tab (2); "% TR. 4:"; p4
130 end
```

Să executăm programul și să analizăm împreună rezultatele (vezi fig. 3.23a).

Eliminarea erorilor de sintaxă

Remarcați pentru început apariția mesajului de eroare:

Syntax error in 20

care ne spune că în linia 20 s-a strecurat o greșeală. Ați găsit-o? Greșeala constă în faptul că s-a utilizat ca separator ";" în loc de ":", ceea ce contravine regulilor de sintaxă prezentate. Se va înlocui așadar simbolul ";" cu ":" în cele două apariții de pe linia 20.

Vom relua execuția programului începînd cu linia 20 (vezi fig. 3.23(b). Încă o eroare de sintaxă! De astă dată în linia 30. Interesant cît de constanți am putut fi săvîrșind aceeași greșeală! Ca și în cazul precedent, se va înlocui simbolul ";" cu ":".

Reluați execuția programului începînd cu linia 30 (vezi fig. 3.23(c).

Chiar să fim urmăriți de ghinion? Nu se poate! De această dată și-au făcut apariția două mesaje:

Division by zero
Syntax error in 40

dintre care primul pare chiar foarte ciudat. Ce s-a întîmplat! Tot corectînd programul am uitat să reluăm execuția programului începînd cu linia 5 și nu cu liniile 20, respectiv 30, cum greșit am procedat. Executînd programul de la linia 30, era normal ca t să fie 0 (numitorul din instrucțiunile de atribuire de la linia 40), deoarece și valorile variabilelor de intrare erau

```
run
a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3
? 10, 20, 30, 5, 6, 7, 15, 20, 25, 30, 5, 8
Syntax error in 20
20 t1= a1+a2+a3 ; t2=b1+b2+b3; t3=c1+c2+c3
```

a)

```
run 20
Syntax error in 30
30 t4= d1+d2+d3; t=t1+t2+t3+t4
```

b)

```
run 30
Division by zero
Syntax error in 40
40 p1 = t1*100/t; p2=t2*100/t;p3=t3*100/t
```

c)

```
run
JUCARII :  TR.1  TR.2  TR.3  TR.4
            60    18    60    43

% TR.1:  33.1491713
% TR.2:   9.94475138
% TR.3:  33.1491713
% TR.4:  43.1491713
Ready
■
```

d)

Fig. 3.23 a–d

```
run
JUCARII: TR.1 TR.2 TR.3 TR.4
          60   18   60   43

% TR.1: 33.1491713
% TR.2:  9.94475138
% TR.3: 33.1491713
% TR.4: 23.7569061

Ready
■
```

Fig. 3.23 e)

zero (neexecutîndu-se instrucţiunile 5 şi 10). În acest caz, fără să vrem, am făcut o greşeală de logică!

În consecinţă, tastaţi:

run [RETURN]

nu înainte însă de a corecta aceeaşi greşeală de sintaxă care ne-a urmărit pe tot parcursul execuţiei programului din linia 40.

În fig. 3.23 (d) se prezintă rezultatul final al execuţiei programului. Foarte mulţi dintre dvs. ar fi tentaţi să se declare mulţumiţi de răspunsul calculatorului manifestînd o grabă sau chiar, în cele mai multe cazuri, o satisfacţie, motivînd că programul „a mers" iar lista cu rezultate este încîntătoare! Desigur, rezultatele sînt tipărite frumos, dar, ştiţi dvs. cum este cu frumuseţea!

Aşadar, vă rugăm să rămînem în continuare asupra listei cu rezultate, propunîndu-vă să faceţi un calcul simplu: adunaţi cele patru procente! Aţi obţinut 100? Nu?! Care credeţi că este cauza? Am făcut pe undeva vreo greşeală de logică? Încercaţi s-o depistaţi singuri.

Eliminarea unei erori de logică

Există mai multe tehnici de depistare a unei erori de logică. Într-un fel, ele ţin şi de practica pe care o aveţi dvs. în programarea calculatoarelor. În ceea ce ne priveşte, pentru problema în cauză vă propunem să parcurgeţi următorii paşi: a) lecturaţi încă o dată problema; b) revedeţi specificaţiile de programare şi documentaţia de proiectare; c) revedeţi lista programului sursă BASIC, linie cu linie.

Aţi găsit greşeala? Nu vă descurajaţi atît de uşor. Sînteţi de-abia la început de drum!

La o primă impresie, se pare că eroarea ar proveni din algoritmul de calcul al valorilor lui p_1, p_2, p_3 şi p_4. Verificaţi în program relaţiile de calcul (vezi liniile 40 şi 50).

```
40 p1=t1 * 100/t : p2=t2 * 100/t : p3=t3 * 100/t
50 p4=t4 * 100/100
```

Mulţi dintre dvs. ar putea afirma că nici aici (liniile 40 şi 50) nu există greşeală, furaţi poate de faptul că expresiile lui p1, p2 şi p3 sînt corecte. Dar p4? Tocmai aici s-a strecurat greşeala. Din neatenţie, în loc să tastăm (la numitor) **t**,

50 p4=t4 * 100/100

am introdus 100, ceea ce reprezintă cu totul altceva!

În consecinţă, corectaţi linia 50 (tastaţi **edit** 50) de forma:

50 p4=t4 * 100/t

tastaţi:

run [RETURN]

pregătindu-vă de o nouă conversaţie (vezi fig. 3.23(e)), sperăm şi ultima pentru această problemă.

Remarci

- Valorile variabilelor de intrare au fost şterse de pe ecran (vezi instrucţiunea **cls** din linia 60), înainte de afişarea capului de tabel şi a rezultatului final;
- Introduceţi şi executaţi programul şi pe calculatoarele aMIC (nu folosiţi liniile multiinstrucţiune), PRAE, HC-85, TIM S, SPECTRUM, COMMODORE, AMSTRAD, M 118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX C, calculatorul dvs., cu recomandarea de a introduce în linia 40 instrucţiunea corespunzătoare, după cum urmează: **INIT** (aMIC), **CLS** (PRAE, HC-85, TIM S SPECTRUM), **PRINT CHR$** (147), (COMMODORE), **CHR$ 24** (M 118, TPD, JUNIOR).

☐ **Programul pentru calculul procentelor consumurilor trimestriale** ale unui articol este:

```
5 print c1 : print c2
10 input c1, c2, c3, c4
20 l=c1+c2+c3+c4:
   p1=c1 * 100/t: p2=c2 * 100/t
30 p3=c3 * 100/t : p4=c4 * 100/t
40 cls
50 print "CONSUMURI:";
   tab (12); "TR. 1"; tab (17); "TR. 2";
   tab (22); "TR. 3"; tab (27); "TR. 4
"
60 print tab (12); c1; tab (17); c2;
   tab (22); c3; tab (27); c4
70 print : print
80 print tab (4); "% TR. 1:"; p1
90 print tab (4); "% TR. 2:"; p2
100 print tab (4); "% TR. 3:"; p3
110 print tab (4); "% TR. 4:"; p4
120 end
```

Introduceţi şi executaţi programul.

Cum vă explicaţi rezultatele pe care le-aţi obţinut?

Să începem cu apariţia celor două zerouri. Ele sînt rezultatul execuţiei instrucţiunii 5:

5 **print** c1 : **print** c2

care, aşa cum a fost scrisă, este inutilă (c1, c2 au, pentru început, valoarea zero). Corectaţi instrucţiunea de forma:

5 **print** "c1, c2, c3, c4"

Să mergem mai departe. Deşi aţi introdus valori pentru variabilele c1, c2, c3 şi c4, calculatorul v-a semnalat de patru ori acelaşi mesaj:

Division by zero.

```
run
0
0
? 321,432,766,876
Division by zero
Division by zero
Division by zero
Division by zero
```

a)

```
CONSUMURI : TR.1  TR.2  TR.3  TR.4
            321   432   766   876

% TR.1 : 13.4029228
% TR.2 : 18.0375783
% TR.3 : 31.9832985
% TR.4 : 36.5762004
Ready
█
```

b)

Fig. 3.24 a–b.

Credem că ați învățat din greșelile conversației precedente și, ca urmare nu v-a fost greu să observați eroarea din linia 20:

```
20 l=c1+c2+c3+c4 : p1=c1 * 100/t : p2=c2 * 100/t
```

în loc de l= . . . trebuie introdus:

```
20 t=c1+c2+c3+c4 : . . .
```

Odată operate aceste două corecții, încercați o nouă execuție a programului. În fig. 3.24 (b) se prezintă rezultatul corect al execuției acestui program. Nu uitați să verificați . . . calculatorul!

Observație. Introduceți și executați programul și pe calculatoarele aMIC (nu folosiți liniile multiinstrucțiune), PRAE, HC-85, TIM-S, SPECTRUM, COMMODORE, AMSTRAD, M-118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX-C, cu recomandarea de a introduce în linia 40 instrucțiunea corespunzătoare fiecărui calculator în parte.

CONVERSAȚIA 4

Proiectarea și realizarea unui program pentru calculul cantității de lichid din nouă rezervoare cilindrice echilaterale a căror rază variază de la 2 la 10 m, cu pasul de un metru. Constante și variabile alfanumerice. Structura de iterație PENTRU. Bucla FOR–NEXT. „Bunicul" lui FOR–NEXT. Editarea valorilor numerice. Instrucțiunea PRINT USING. Modificatorul FOR. APLICAȚII și TESTE pentru cititor



□ De la problemă la program

Vom relua problema precedentă considerînd de această dată mai multe rezervoare cilindrice echilaterale (generatoarea este egală cu diametrul), toate pline cu benzină! Raza acestora variază de la 2 la 10 m cu pasul 1 (2 m, 3 m, . . . ,10 m). Problema pe care ne-o propunem să o rezolvăm acum constă în a calcula şi afişa cantitatea de benzină în tone (masa) din fiecare rezervor. De asemenea, în final se doreşte să se afişeze cantitatea totală de benzină. Densitatea benzinei (0,7 g/cm^3) se va introduce în momentul execuţiei programului. Cum s-ar putea rezolva această problemă?

Cei mai grăbiţi dintre dvs. se şi gîndesc deja la instrucţiunile BASIC necesare alcătuirii programului. Cîteva **INPUT**-uri pentru citirea datelor de intrare, cîteva instrucţiuni de atribuire necesare calculului cantităţii de benzină din fiecare rezervor, o instrucţiune care să însumeze cantitatea de benzină pe care o conţin cele 9 rezervoare şi cîteva **PRINT**-uri! Banal, vor exclama poate mulţi dintre aceştia. A! vor întîmpina poate o „singură" dificultate privind formula densităţii unui corp. Le-o vom aminti noi:

$$\rho = \frac{m}{V} \qquad (1)$$

unde ρ reprezintă densitatea, m masa corpului, iar V volumul cilindrului (aria bazei înmulţită cu generatoarea).

În ceea ce ne priveşte, nu recomandăm rezolvarea informatică a problemei propuse în acest mod defectuos în care gîndesc foarte mulţi debutanţi. A gîndi în limbaj informatic (BASIC) încă de la formularea problemei ni se pare o concepţie total greşită. Rezolvarea în grabă a problemei este urmată de cele mai multe ori de surprize – evident neplăcute. Spre a le evita vă propunem să străbatem împreună *lungul drum al problemei către program*, prin parcurgerea tuturor fazelor evidenţiate în metodica de analiză, proiectare şi realizare a programelor.

□ Analiza problemei

Date de ieşire, intrare şi stare

Şi în rezolvarea acestei probleme pornim cu datele de ieşire, urmînd să ne întrebăm apoi dacă avem suficiente date de intrare pentru a putea fi obţinute ieşirile dorite. Se urmăreşte ca raportul de ieşire să aibă formatul descris în modulul de analiză structurată, fig. 4.1 (a) în care, rîndul de titlu (RAZA, MASA) este urmat de valorile razelor (2, 3, . . . ,10m) rezervoarelor (cilindrice echilaterale) şi de cantităţile de benzină corespunzătoare (calculate prin program) şi în final de un rînd de TOTAL în care se afişează cantitatea de benzină din cele 9 rezervoare.

Datele de intrare cuprind numai densitatea benzinei care se va citi (dinamic) în timpul execuţiei programului. Este momentul să ne întrebăm dacă datele de ieşire (M, S) şi de intrare (D) sînt suficiente pentru rezolvarea problemei! Să nu ne grăbim cu răspunsul şi să analizăm cu mai mare atenţie formatul datelor de ieşire (vezi fig. 4.1(a)). Vă propunem să ne oprim chiar asupra primului element al listei: RAZA – care ia valori într-un domeniu cunoscut [2, 10] m. Aşadar, nu ne rămîne decît să-i asociem acestei date un nume de variabilă (R) pe care să o definim ca o *variabilă de stare.*

```
= = = = = = = = = = = = = = = = = = = = = = = = =
RAZA                          .MASA
= = = = = = = = = = = = = = = = = = = = = = = = =
 2                            xxx.xx
 3                            xxx.xx
 .                              .
 .                              .
 10                           xxx.xx
= = = = = = = = = = = = = = = = = = = = = = = = =
TOTAL =                       xxx.xx TONE
= = = = = = = = = = = = = = = = = = = = = = = = =
```

a)

TABELA DE VARIABILE

Nume program: EXEMPLELE 4 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieşire
D: densitatea benzinei	V: volumul rezervorului cilindric echilateral R: raza rezervorului M: masa de benzină	S: masa totală de benzină M: masa de benzină

b)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 4 – PC, m, M, F

Descriere. Programul calculează şi afişează cantitatea de benzină din mai multe rezervoare cilindrice echilaterale (generatoarea este egală cu diametrul) a căror rază variază de la 2 la 10 metri cu pasul de un metru.

Intrări. Se citeşte dinamic densitatea benzinei (0,7 g/cm³).

Ieşiri. Masa de benzină din fiecare rezervor şi masa totală de benzină.

Lista de funcţiuni ale programului

1. Citeşte densitate benzină
2. Iniţializează cu zero variabila S
3. Tipăreşte cap tabel
4. Calculează volumul rezervorului
5. Calculează masa de benzină (M) din fiecare rezervor
6. Însumează M în S
7. Afişează raza (R) şi masa rezervorului (M)
8. Iniţializează variabila de control a ciclului
9. Incrementează şi testează variabila de control
10. Afişează masa totală de benzină
11. Stop

c)

Fig. 4.1. a–d. Modulul de analiză structurată. *a)* formatul datelor de ieşire; *b)* tabela de variabile; c) specificaţii de programare;

TABELA DE VARIABILE

Nume program: EXEMPLELE 4 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieşire
	S: masa totală de benzină	

d)

d) tabela de variabile extinsă.

În tabela de variabile a EXEMPLELOR 4 – PC, m, M, F s-au definit încă două variabile de stare: V, M a căror semnificaţie este redată în modulul de analiză structurată, fig. 4.1 (b).

Descrierea programului, precum şi lista principalelor funcţiuni sînt prezentate în modulul de analiză structurată, fig. 4.1 (c).

Observaţie. În cadrul acestei conversaţii, prelucrările se fac nu pentru un rezervor (articol), ci pentru mai multe (9) rezervoare (articole)!

Asigurîndu-ne că nu s-au strecurat greşeli în faza de analiză a problemei, putem aborda în continuare faza de proiectare a programului urmînd să elaborăm în cele ce urmează: diagrama de structură, pseudocodul şi schema logică.

□ Proiectarea programului

Proiectarea diagramei de structură

În vederea trasării diagramei de structură vom proceda mai întîi la alocarea funcţiunilor de prelucrare (vezi modulul de analiză structurată fig. 4.1 (c)) unor module, după cum urmează: CAP–TAB, INIT–CIT, CALCUL, EDIT–RIND, TOTAL, SFÎRŞIT (vezi modulul de proiectare structurată, fig. 4.2 (a)).

Modulele CAP–TAB, INIT–CIT, TOTAL, SFÎRŞIT împreună cu modulul PRELUCRĂRI, pe care l-am introdus în mod artificial, alcătuiesc nivelul 1 al diagramei de structură desenate în fig. 4.2 (b).

Modulele CALCUL şi EDIT–RÎND, ce aparţin noului modul introdus, alcătuiesc nivelul 2 al diagramei de structură.

De notat că, nu întotdeauna primul nivel al unei diagrame de structură satisface toate cerinţele de prelucrare a programului. De aceea, apare ca necesară rafinarea (detalierea) anumitor blocuri (vezi blocul PRELUCRĂRI).

După cum remarcaţi, blocul PRELUCRĂRI conţine în colţul din dreapta sus un asterisc care are următoarea semnificaţie: «EXEMPLUL 4 "constă din", mai multe PRELUCRĂRI ce "constau din" CALCUL "urmat de" EDIT–RÎND».

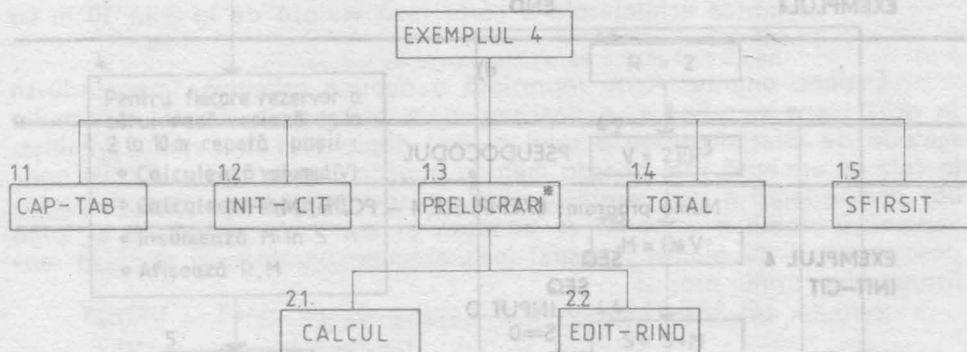
Altfel spus, blocul PRELUCRĂRI reprezintă o structură de repetiţie (iteraţie), în cazul nostru cu număr cunoscut de paşi –9.

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE

Nume program: ELEMPLELE 4 – PC, m, M, F

Modul	Funcţiuni
CAP–TAB	3
INIT–CIT	1, 2
CALCUL	4, 5, 6
EDIT–RIND	7
TOTAL	10
SFÎRŞIT	11
PRELUCRĂRI	8, 9

a)



b)

PSEUDOCODUL

Nume program: EXEMPLELE 4 – PC, m, M, F

B : citeşte densitatea benzinei
C : afişează cap-tabel

PENTRU fiecare rezervor a cărui rază variază **DE LA** 2 **LA** 10

A D
D"1 : calculează volumul
D"2 : calculează masa de benzină
D"3 : însumează masa de benzină calculată în cantitatea totală de benzină
D"4 : afişează raza şi masa de benzină

SFÎRŞIT

E : afişează masa totală de benzină

c)

Fig. 4.2. Modulul de proiectare structurată: *a)* alocarea funcţiunilor de prelucrare; *b)* diagrama de structură; *c)* pseudocodul (varianta 1).

PSEUDOCODUL

Nume program: EXEMPLELE 4 – PC, m, M, F

```
EXEMPLUL4      SEQ
INIT–CIT         SEQ
                   INPUT D
                   Inițializare variabile
INIT–CIT         END
PRELUCRĂRI       PENTRU R DE LA 2 LA 10
CALCUL             SEQ
                     V=2 π R³
                     M=D * V
                     S=S+M
CALCUL             END
                   PRINT R, M
PRELUCRĂRI       SFÎRȘIT
                 PRINT S
EXEMPLUL4      END
```

d)

PSEUDOCODUL

Nume program: EXEMPLELE 4 – PC, m, M, F

```
EXEMPLUL 4     SEQ
INIT–CIT         SEQ
                   INPUT D
                   S=0
INIT–CIT         END
```

(se completează cu blocul PRELUCRĂRI din fig. 4.2 (d)

.

.

e)

d) pseudocodul (varianta 2); e) pseudocodul (varianta finală);

Test

Verificați dacă modulele „CALCUL" și „EDIT-RÎND" se repetă de nouă ori.

De notat că o astfel de reprezentare (de tip JACKSON) se cere a fi completată cu informații (structurate de obicei într-un tabel) suplimentare cu rolul de a specifica în clar toate condițiile ce intervin în cadrul programului.

Remarcă. **Numele blocurilor ce alcătuiesc diagrama de structură trebuie să fie cît mai sugestive, scurte, reprezentative.**

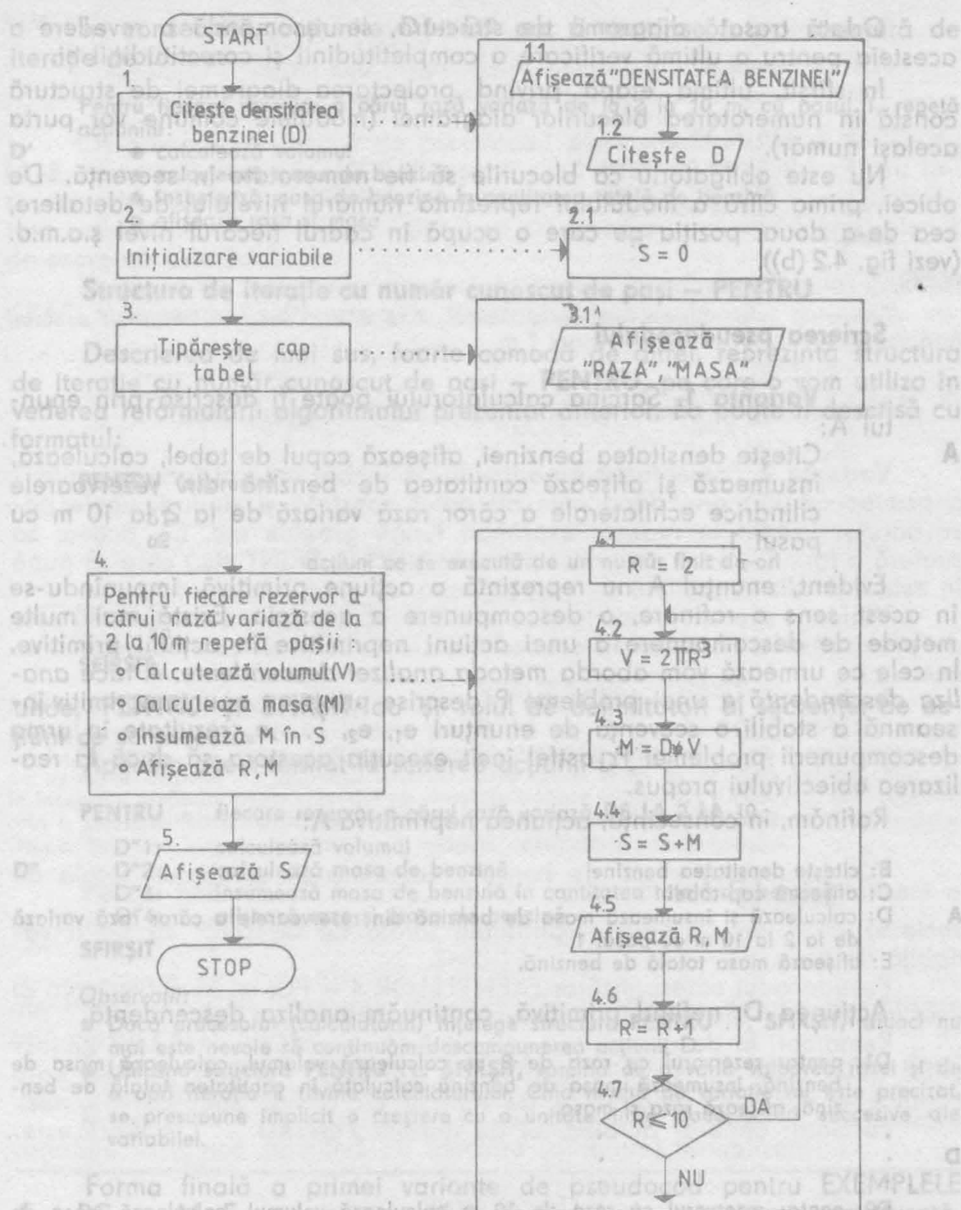


Fig. 4.2. *f*) schema logică.

Nu puţine sînt cazurile cînd, pentru prelucrarea unui articol, folosim un modul comun şi altor prelucrări.

Din analiza figurii 4.2 (b) ce conţine modulele de prelucrare cu funcţiunile corespunzătoare, se observă că pentru EXEMPLELE 4 – PC, m, M, F nu întîlnim module cu funcţiuni comune.

Odată trasată diagrama de structură, se recomandă o revedere a acesteia pentru o ultimă verificare a completitudinii şi corectitudinii ei.

În sfîrşit, ultima etapă privind proiectarea diagramei de structură constă în numerotarea blocurilor diagramei (modulele comune vor purta acelaşi număr).

Nu este obligatoriu ca blocurile să fie numerotate în secvenţă. De obicei, prima cifră a modulului reprezintă numărul nivelului de detaliere, cea de-a doua, poziţia pe care o ocupă în cadrul fiecărui nivel ş.a.m.d. (vezi fig. 4.2 (b)).

Scrierea pseudocodului

Varianta 1. Sarcina calculatorului poate fi descrisă prin enuntul A:

A Citeşte densitatea benzinei, afişează capul de tabel, calculează, însumează şi afişează cantitatea de benzină din rezervoarele cilindrice echilaterale a căror rază variază de la 2 la 10 m cu pasul 1.

Evident, enunţul A nu reprezintă o acţiune primitivă, impunîndu-se în acest sens o rafinare, o descompunere a acesteia. Există mai multe metode de descompunere a unei acţiuni neprimitive în acţiuni primitive. În cele ce urmează vom aborda *metoda analizei descendente*. A face *analiza descendentă* a unei probleme P descrise printr-un enunţ neprimitiv înseamnă a stabili o secvenţă de enunţuri e_1, e_2, . . . ,e_n rezultate în urma descompunerii problemei P, astfel încît execuţia acestora să ducă la realizarea obiectivului propus.

Rafinăm, în consecinţă, acţiunea neprimitivă A:

A
B: citeşte densitatea benzinei
C: afişează cap-tabel
D: calculează şi însumează masa de benzină din rezervoarele a căror rază variază de la 2 la 10 m cu pasul 1
E: afişează masa totală de benzină.

Acţiunea D, nefiind primitivă, continuăm analiza descendentă.

D
D1: pentru rezervorul cu raza de 2 m, calculează volumul, calculează masa de benzină, însumează masa de benzină calculată în cantitatea totală de benzină, afişează raza şi masa.
. .
. .
. .
D9: pentru rezervorul cu raza de 10 m calculează volumul, calculează masa de benzină, însumează masa de benzină calculată în cantitatea totală de benzină, afişează raza şi masa.

Se observă că acţiunile D1÷D9 se repetă pentru valori ale razei cuprinse între 2 şi 10 m. Apare deci necesară existenţa unei structuri algoritmice care să permită descrierea repetiţiei (iteraţiei) într-o formă comodă. O astfel de structură care defineşte un grup de acţiuni (enunţuri) sau o combinaţie de structuri fundamentale algoritmice cu execuţie repetată este cunoscută sub denumirea de *structură iterativă*.

În consecinţă, acţiunile D1÷D9 pot fi restrînse într-o structură de iteraţie de forma:

D' Pentru fiecare rezervor a cărui rază ,variază de la 2 la 10 m, cu pasul 1, repetă acţiunile:
- calculează volumul
- calculează masa de benzină
- însumează masa de benzină în cantitatea totală de benzină
- afişează raza şi masa

Structura de iteraţie cu număr cunoscut de paşi – PENTRU

Descrierea de mai sus, foarte comodă de altfel, reprezintă structura de iteraţie cu număr cunoscut de paşi – **PENTRU,** pe care o vom utiliza în vederea reformulării algoritmului prezentat anterior. Ea poate fi descrisă cu formatul:

PENTRU ⟨expresie⟩
a1
a2
· acţiuni ce se execută de un număr finit de ori
·
·
an
SFÎRŞIT

unde, **PENTRU** şi **SFÎRŞIT** au şi rolul de delimitatori ai secvenţei de acţiuni ce se execută.

Aplicăm acest format la scrierea acţiunii D .

PENTRU fiecare rezervor a cărui rază variază **DE LA 2 LA 10**
D″1: calculează volumul
D″ D″2: calculează masa de benzină
D″3: însumează masa de benzină în cantitatea totală de benzină
D″4: afişează raza şi masa de benzină
SFÎRŞIT

Observaţii:

- Dacă procesorul (calculatorul) înţelege structura **PENTRU . . . SFÎRŞIT,** atunci nu mai este nevoie să continuăm descompunerea acţiunii D.
- Utilizînd structura **PENTRU . . . SFÎRŞIT,** sarcina de a varia valoarea razei şi de a opri iteraţia îi revine calculatorului. Cînd modul de variaţie nu este precizat, se presupune implicit o creştere cu o unitate între două valori succesive ale variabilei.

Forma finală a primei variante de pseudocod pentru EXEMPLELE 4 – PC, m, M, F este prezentată în modulul de proiectare structurată, fig. 4.2(c).

Aplicaţie. Un cilindru are înălţimea de 10 cm, iar raza R variabilă. Transcrieţi în pseudocod algoritmul de calcul al ariei laterale, totale şi al volumului cilindrului cînd R variază de la 1 la 18 cm, cu pasul de 1 cm.

Sarcina calculatorului poate fi descrisă prin acţiunea **A**:

A pentru fiecare cilindru a cărui rază variază de la 1 la 18 cm, cu pasul 1, repetă acţiunile: a) calculează şi tipăreşte aria laterală; b) calculează şi tipăreşte aria totală; c) calculează şi tipăreşte volumul.

Se observă că putem identifica o structură de iteraţie de tip **PENTRU**.

PENTRU fiecare cilindru a cărui rază variază **DE LA** 1 **LA** 18

A1′ : calculează aria laterală
A2′ : tipăreşte aria laterală
A A3′ : calculează aria totală
A4′ : tipăreşte aria totală
A5′ : calculează volumul
A6′ : tipăreşte volumul

SFÎRŞIT

TEST

Transcrieţi în pseudocod algoritmul de calcul al ariei şi lungimii cercurilor a căror rază variază de la 10 la 20 m, cu pasul 1 m.

Prezentarea acţiunilor primitive de o manieră formalizată

Varianta 2. Prezentarea algoritmului în limbaj natural (vezi varianta 1 a pseudocodului) are, după cum aţi putut constata şi singuri, unele dezavantaje: lizibilitate redusă, exprimări lungi, greoaie etc. De aceea, se preferă o formalizare a acţiunilor primitive (**INPUT, PRINT** etc.) care să pună în evidenţă atît variabilele de intrare/ieşire, cît şi variabilele de stare.

Varianta prezentării acţiunilor primitive de o manieră formalizată este de preferat celei narative. În consecinţă, s-o folosim cît este cu putinţă pe aceasta!

Înainte de a prezenta noua variantă a pseudocodului corespunzătoare EXEMPLELOR 4 – PC, m, M, F, va trebui să completăm tabela de variabile cu o nouă variabilă de stare – S (vezi modulul de analiză structurată, fig. 4.1(d)).

Această operaţie este necesară deoarece acţiunea primitivă

„
D3: însumează masa de benzină calculată în cantitatea totală de benzină"

reclamă definirea variabilei "cantitatea totală de benzină" ca pe o variabilă de stare.

Pseudocodul corespunzător EXEMPLELOR 4 – PC, m, M, F (varianta 2) este prezentat în modulul de proiectare structurată, fig. 4.2(d).

Remarcaţi în descrierea pseudocodului instrucţiunile standard cunoscute: **SEQ, END** dar şi noile instrucţiuni standard: **PENTRU, SFÎRŞIT** care codifică o structură algoritmică de iteraţie cu număr cunoscut de paşi. Formatul general al blocului de iteraţie **PENTRU . . . SFÎRŞIT** este:

Format general		
Etichetă	**PENTRU**	**V DE LA** V1 **LA** V2 **PAS** R
	(corpul buclei)	
Etichetă	**SFÎRŞIT**	

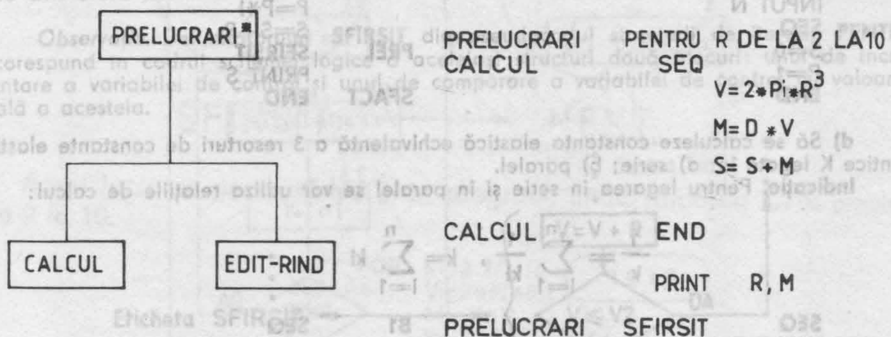
unde, V este variabila de control, V1 valoarea iniţială a variabilei, V2 valoarea finală a structurii de iteraţie, R pasul de incrementare.

REMARCI

- Instrucţiunile **PENTRU** şi **SFÎRŞIT** au obligatoriu aceeaşi etichetă.
- Instrucţiunea **PENTRU** reprezintă punctul unic de intrare în bloc.
- Instrucţiunea **SFÎRŞIT** reprezintă punctul unic de ieşire din bloc.

Cît priveşte varianta 2 de pseudocod pentru EXEMPLELE 4 – PC, m, M, F, remarcaţi cum PRELUCRĂRI joacă rolul etichetei (aceeaşi pentru instrucţiunile **PENTRU** şi **SFÎRŞIT**, iar acţiunile cuprinse între cei doi delimitatori reprezintă acţiuni (instrucţiuni) ce se execută în secvenţă (vezi blocul de secvenţă CALCUL).

În figura de mai jos se prezintă în paralel imaginea blocului PRELUCRĂRI reprezentat în diagrama de structură şi în pseudocod.



Remarcă. Nu putem trece neobservată acţiunea primitivă: „Iniţializare variabile" ce aparţine blocului de secvenţă INIT-CIT. Este foarte greu să ne dăm seama chiar de la începutul alcătuirii pseudocodului ce variabile trebuie iniţializate. De aceea, vom proceda prin a semnala într-un prim pas acţiunea de iniţializare a variabilelor, urmînd să revenim atunci cînd am obţinut deja o primă imagine completă a pseudocodului.

Pentru EXEMPLELE 4 – PC, m, M, F, singura cerinţă de iniţializare o reclamă acţiunea (instrucţiunea):

S=S+M

din blocul CALCUL (aparţine structurii algoritmice de iteraţie identificate prin blocul PRELUCRĂRI).

Evident, execuţia acţiunii "S=S+M" nu are sens pentru noi decît dacă S a primit în prealabil o valoare iniţială, respectiv zero.

În consecinţă, vom introduce în locul acţiunii neprimitive „Iniţializare variabile" acţiunea primitivă de afectare (atribuire) formalizată:

S=0

Varianta 2 a pseudocodului, completă este prezentată în modulul de proiectare structurată, fig. 4.2(e).

Aplicaţii. Alcătuiţi pseudocodul pentru rezolvarea următoarelor probleme:

a) Să se calculeze suma primelor n numere naturale.

```
A  SEQ                          A  SFÎRŞIT
   INPUT N                         PRINT S
   S=O                          B  END
B  PENTRU I DE LA 1 LA N
   S=S+I
```

TEST

Modificaţi algoritmul de mai sus pentru calculul sumei pătratelor primelor n numere naturale.

b) Să se calculeze produsul primelor n numere naturale (calculul lui n!)

```
FACT  SEQ                          P=P*I
      INPUT N                F     SFÎRŞIT
      P=1                          PRINT P
F     PENTRU I DE LA 1 LA N  FACT  END
```

c) Să se calculeze $\sum_{K=1}^{n} K!$

```
SFACT SEQ                    PREL  PENTRU I DE LA 1 LA N
      INPUT N                      P=P*I
S2    SEQ                          S=S+P
      S=0                    PREL  SFÎRŞIT
      P=1                          PRINT S
S2    END                    SFACT END
```

d) Să se calculeze constanta elastică echivalentă a 3 resorturi de constante elastice identice K legate în: a) serie; b) paralel.

Indicaţie. Pentru legarea în serie şi în paralel se vor utiliza relaţiile de calcul:

$$\frac{1}{k}=\sum_{i=1}^{n}\frac{1}{ki},\quad k=\sum_{i=1}^{n} ki$$

```
A1    SEQ                    B1    SEQ
      INPUT K                      S=S+1/K
      S=0                          P=P+K
      P=0                    B1    END
B     PENTRU I DE LA 1 LA 3  B     SFÎRŞIT
                                   PRINT P, S
                             A1    END
```

Trasarea schemei logice

Schema logică pentru EXEMPLELE 4 – PC, m, M, F este reprezentată în modulul de proiectare structurată, fig. 4.2(f). Ne vom opri în cele ce urmează asupra modului de reprezentare grafică a structurii de repetiţie cu

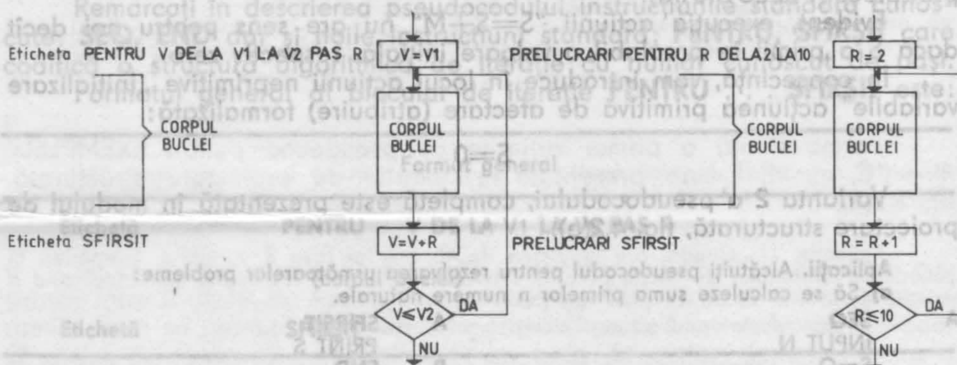
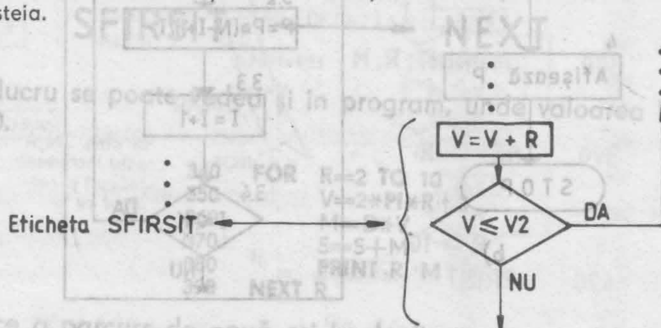


Fig. 4.3.

număr fínit de paşi **PENTRU . . . SFÎRŞIT** (vezi blocul 4 din fig. 4.2(f) şi rafinarea acestuia în 4.1, . . ., 4.7). Pentru o înţelegere mai uşoară a mecanismului privind trasarea schemei logice vom apela din nou la reprezentarea în pseudocod (vezi fig. 4.3) a structurii **PENTRU.**

După cum constataţi, schema logică a structurii de iteraţie **PENTRU** începe cu un bloc de iniţializare a variabilei de control (V=V1), continuă cu corpul buclei şi se termină cu un bloc de incrementare a variabilei de control V (V=V+R), urmat de un bloc de decizie. În cadrul blocului de decizie se testează valoarea variabilei V după cum urmează: dacă valoarea lui V este mai mică sau egală cu V2 (valoarea finală a variabilei V) se reia ciclul începînd cu primul bloc de după V=V1, în caz contrar se iese din ciclu urmînd a se executa blocul imediat următor.

Observaţie. Delimitatorului **SFÎRŞIT** din pseudocodul structurii de iteraţie **PENTRU** îi corespund în cadrul schemei logice a aceleiaşi structuri două blocuri: unul de incrementare a variabilei de control şi unul de comparare a variabilei de control cu valoarea finală a acesteia.



Aplicaţie. Să se calculeze C_n^k. Se va alcătui tabela de variabile şi se va trasa schema logică.

Indicaţie. Se va utiliza formula

$$C_n^k = \frac{n!}{k!\ (n-k)!} \qquad (2)$$

unde $n \geqslant k$, $n > 0$, $k > 0$.

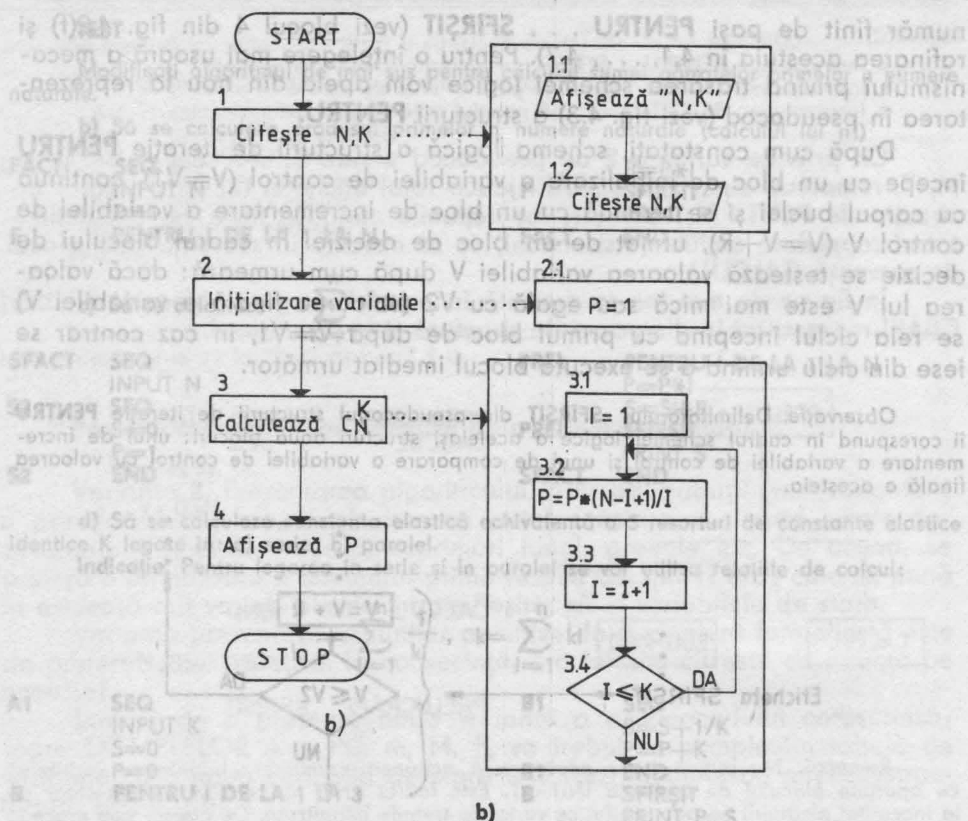
Tabela de variabile şi schema logică a aplicaţiei sînt prezentate în modulele de analiză şi proiectare structurată, fig. 4.4.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieşire
N: numărul de elemente ale mulţimii finite K: numărul submulţimilor avînd fiecare cîte K elemente	I: variabila de control a structurii de iteraţie **PENTRU**	P: numărul combinărilor de n elemente luate cîte k

a)

Fig. 4.4. a-b. *a)* tabela de variabile;



b) schema logică.

□ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 205

FOR – NEXT

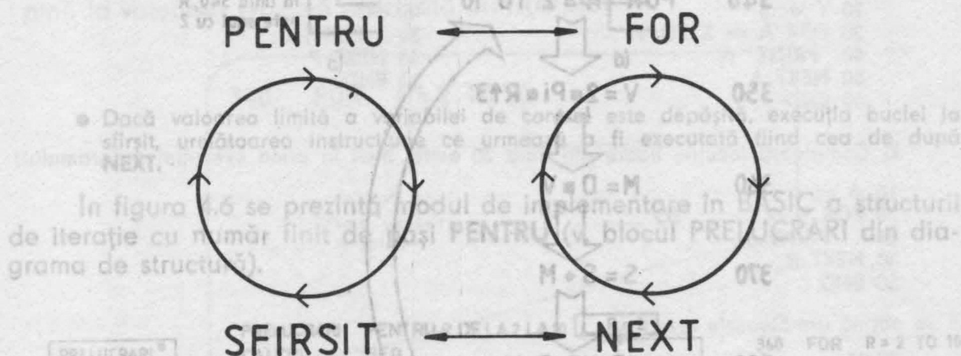
Programul pe care îl analizăm în cele ce urmează conţine unul din cele mai "forte" instrumente de conversaţie BASIC: bucla **FOR – NEXT**.

Cele două instrucţiuni considerate ca fiind cele mai "tari" instrucţiuni ale BASIC-ului, din păcate foarte des confundate de începători, sînt echivalentul instrucţiunilor **PENTRU** respectiv **SFÎRŞIT** din structura de repetiţie cu număr finit de paşi întîlnită la reprezentarea în pseudocod.

Cu instrucţiunile **FOR** şi **NEXT** spunem calculatorului de cîte ori să treacă prin buclă. După aceea, la ieşirea din buclă, calculatorul continuă execuţia programului cu instrucţiunea imediat următoare.

Instrucţiunile **FOR** şi **NEXT** apar în linii diferite în cadrul programului. **FOR** reprezintă punctul de intrare în buclă şi apare întotdeauna prima, iar **NEXT** este ultima instrucţiune din buclă reprezentînd punctul de ieşire din ciclu (buclă). Cît priveşte instrucţiunea sau instrucţiunile din buclă, cu-

prinse între **FOR** şi **NEXT**, acestea sînt executate în mod repetat în ordinea numerelor de linie, de atîtea ori de cît s-a precizat în instrucţiunea **FOR**.



Acest lucru se poate vedea şi în program, unde valoarea lui R creşte de la 2 la 10.

```
340  FOR  R=2 TO 10
350       V=2*PI*R↑3
360       M=D*V
370       S=S+M
380       PRINT R, M
390  NEXT R
```

După ce a parcurs de nouă ori bucla (instrucţiunea **FOR** spune calculatorului ca R să ia valori de la 2 la 10) calculatorul execută instrucţiunea 420, imediat următoare, după **NEXT** (400 este instrucţiune neexecutabilă) şi în final execută instrucţiunile 430 şi 440.

Observaţie. Nu este obligatoriu ca R să ia valori incepînd cu 1!

TEST

În programul precedent, bucla **FOR-NEXT** ocupă liniile .... şi ....

R. 340, 350, 360, 370, 380, 390.

Cum lucrează bucla FOR–NEXT?

Urmăriţi săgeţile din figura 4.5. După cum constataţi, calculatorul ajunge de fiecare dată la instrucţiunea **NEXT** R, incrementează valoarea lui R cu o unitate şi compară noua valoare cu 10 (valoarea limită a lui R). Cînd valoarea lui R este mai mare decît 10, calculatorul continuă execuţia programului cu următoarea instrucţiune din program, care este în afara buclei.

Aţi înţeles? Să vedem. Prima valoare din buclă a lui R este 2 (R=2). La a doua parcurgere a buclei

R=R+1=2+1=3

la a opta parcurgere a buclei R=?

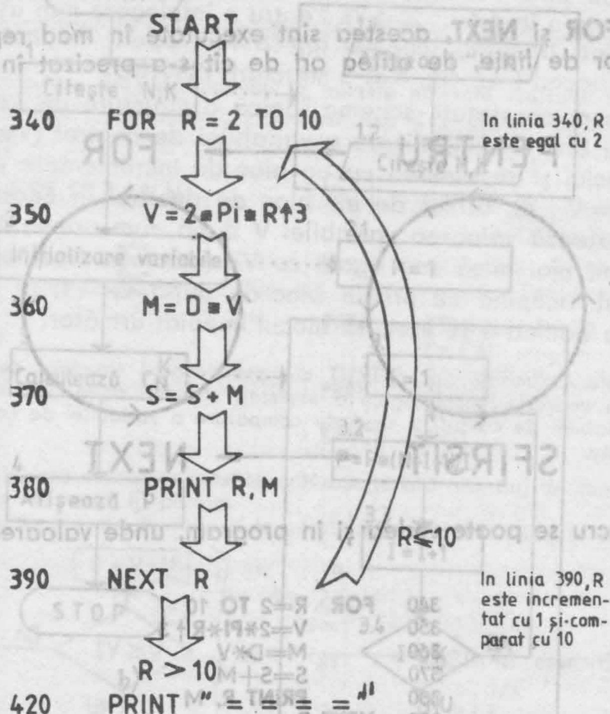


Fig. 4.5. Mecanismul de funcţionare a buclei **FOR-NEXT**.

Dacă aţi răspuns 9, felicitări, dacă nu vă invităm să citiţi încă o dată explicaţiile din fig. 4.5.

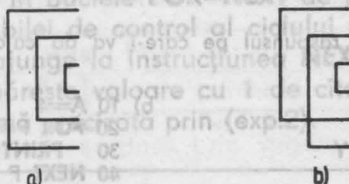
Formatele generale ale instrucţiunilor **FOR** şi **NEXT** sînt:

Format general
⟨nr. linie⟩ **FOR** ⟨variabilă⟩ = ⟨exp.1⟩ **TO** ⟨exp.2⟩ [**STEP** ⟨exp.3⟩]

Format general
⟨nr. linie⟩ **NEXT** ⟨variabilă⟩

Remarci:

- ⟨variabilă⟩ reprezintă o variabilă simplă, de control a buclei.
- ⟨variabilă⟩ din instrucţiunea **NEXT** este aceeaşi cu ⟨variabilă⟩ din instrucţiunea **FOR**.
- ⟨exp.1⟩ este valoarea iniţială care se atribuie variabilei de control a buclei.
- ⟨exp.2⟩ este valoarea finală a variabilei de control a buclei.
- **STEP** este opţional.
- ⟨exp.3⟩ este incrementul (pasul) care se adună la valoarea variabilei de control; în absenţa acestuia se ia automat valoarea unu.
- ⟨exp.1⟩, ⟨exp.2⟩, ⟨exp.3⟩ sînt expresii numerice.
- în interiorul unei bucle **FOR** pot exista maximum trei bucle incluse una în alta cu condiţia ca acestea să nu se intersecteze.



- Dacă valoarea limită a variabilei de control este depăşită, execuţia buclei ia sfîrşit, următoarea instrucţiune ce urmează a fi executată fiind cea de după **NEXT**.

În figura 4.6 se prezintă modul de implementare în BASIC a structurii de iteraţie cu număr finit de paşi **PENTRU** (v. blocul PRELUCRĂRI din diagrama de structură).

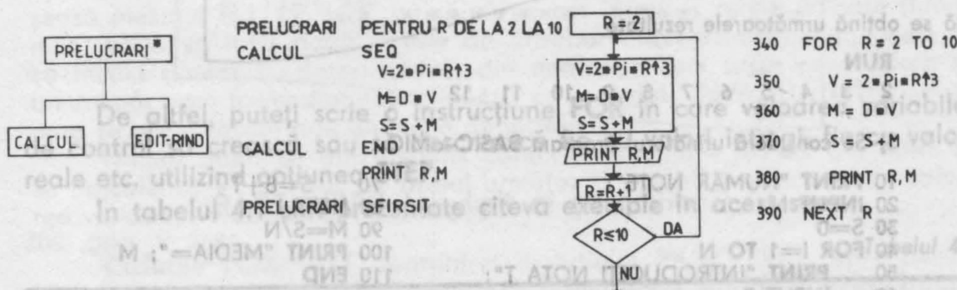


Fig. 4.6

TEST

Scrieţi cîte o buclă **FOR-NEXT** pentru fiecare din cazurile:

a) o buclă trebuie repetată de 999 ori;

b) o buclă trebuie repetată de cîte ori este necesar pentru ca variabila de control să varieze de la 2 la 82, din 4 în 4.

c) o buclă trebuie repetată de cîte ori este necesar pentru ca variabila de control să varieze de la 0.01 pînă la valoarea dată de expresia B*B*B–8. La fiecare execuţie a buclei variabila de control îşi va mări valoarea cu B+K.

Mai multe despre FOR–NEXT

Un alt aspect ce trebuie notat în legătură cu bucla **FOR–NEXT** este acela că atît valoarea iniţială cît şi valoarea finală a variabilei de control pot fi nu numai numere ci şi expresii (v. formatul general al instrucţiunii **FOR**). Aceeaşi observaţie este valabilă şi pentru pasul de incrementare. Iată un exemplu în acest sens:

```
330 S=0
335 A=2
336 B=10
340 FOR R=A TO B
350 V=2*PI*R↑3
360 M=D*V
370 S=S+M
380 PRINT R, M
390 NEXT R
```

Spre a vă convinge de justeţea celor afirmate executaţi programul BASIC-aMIC modificat (v. liniile 335, 336, 340) şi confruntaţi rezultatele. Evident ele vor fi aceleaşi.

TESTE

1) Precizaţi care va fi răspunsul pe care-l va da calculatorul personal aMIC în urma execuţiei programelor:

a)
```
10 X = 0
20 Y = 3
30 FOR A = X TO Y
40  PRINT  A
50 NEXT A
60 END
```

b)
```
10 A=4
20 FOR P=A TO 2*A-1
30   PRINT P;
40 NEXT P
50 END
```

2) Completaţi locurile libere din linia 20 astfel încît în urma execuţiei programului:

```
10 A = 4
20 FOR B = ........... TO ...........
30   PRINT B;
40 NEXT B
50 END
```

să se obţină următoarele rezultate

```
RUN
2  3  4  5  6  7  8  9  10  11  12
```

3) Se consideră următorul program BASIC–aMIC

```
10 PRINT "NUMĂR NOTE";
20 INPUT N
30 S=0
40 FOR I=1 TO N
50    PRINT "INTRODUCEŢI NOTA T";
60    INPUT T
70    S=S+T
80 NEXT I
90 M=S/N
100 PRINT "MEDIA="; M
110 END
```

a) Realizaţi singuri specificaţiile de programare şi documentaţia de proiectare a acestui program.
b) În programul de mai sus bucla **FOR–NEXT** ocupă liniile ....
c) Care linie din bucla **FOR–NEXT** calculează suma notelor (T)?
d) Dacă N=4, de cîte ori va fi afişat pe ecran "INTRODUCEŢI NOTA T"?

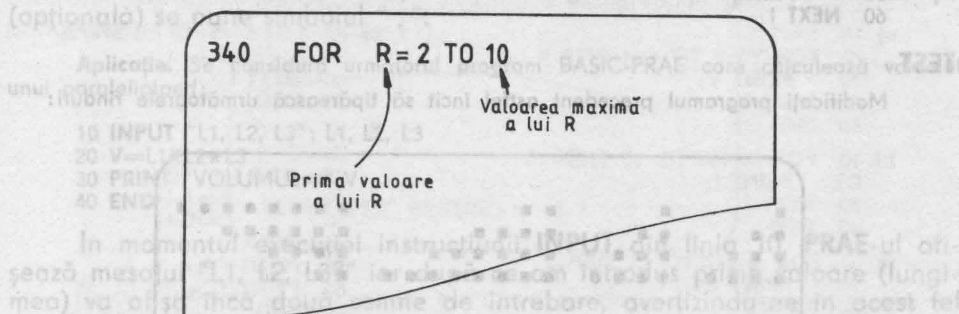
4) Următorul program realizează produsul a N numere! Completaţi programul de mai jos cu instrucţiunile care lipsesc:

```
10 PRINT "N= ";
20 INPUT N
30 PRINT
40 PRINT
50 ............ ?
60 ............ ?
70 PRINT "A= ";
80 INPUT A
90 ............ ?
100 NEXT I
110 PRINT
120 PRINT "PRODUSUL= "; P
130 END
```

Rezultatul execuţiei programului este:

```
RUN
N=?4
A=?20
A=?10
A=?2
A=?3
PRODUSUL = 1200
```

S-a observat că în buclele **FOR–NEXT** de pînă acum, prima valoare care se atribuie variabilei de control al ciclului prin ⟨exp.1⟩ se păstrează pînă ce calculatorul ajunge la instrucţiunea **NEXT**. Apoi, variabila din instrucţiunea **FOR** îşi măreşte valoare cu 1 de cîte ori se reparcurge bucla pînă la valoarea maximă precizată prin ⟨exp.2⟩.



De altfel, puteţi scrie o instrucţiune **FOR** în care valoarea variabilei de control să crească sau să descrească fie cu valori întregi, fie cu valori reale etc. utilizînd opţiunea **STEP**.

În tabelul 4.1 sînt prezentate cîteva exemple în acest sens.

Tabelul 4.1

Instrucţiune	Semnificaţie
10 **FOR** I=1 **TO** 8 **STEP** 2	Valoarea lui I creşte cu 2 ori de cîte ori se parcurge bucla cu condiţia ca I⩽8.
10 **FOR** A=4 **TO** 7 **STEP** 1.5	Valoarea lui A creşte cu 1.5 ori de cîte ori se parcurge bucla cu condiţia ca A⩽7.
10 **FOR** X=25 **TO** 5 **STEP** – 1	Valoarea lui X descreşte cu 1 ori de cîte ori se parcurge bucla pînă ce X este mai mic decît 5. *Observaţie.* X porneşte de la 25 şi ajunge la 5!

"Bunicul" lui FOR–NEXT

Nu puţine sînt cazurile cînd în interiorul unei bucle **FOR** trebuie să facem loc unei alte bucle **FOR** ş.a.m.d. Spunem că avem de-a face cu cicluri în cicluri sau cicluri suprapuse (cicluri imbricate). După cum precizam (v. formatul general al instrucţiunii) condiţia fundamentală de definire corectă a ciclurilor imbricate (maximum trei în BASIC-aMIC) este neintersecţia acestora. De notat, că fiecare ciclu trebuie să aibă propriile instrucţiuni **FOR, NEXT** şi propriile variabile de control ale buclelor.

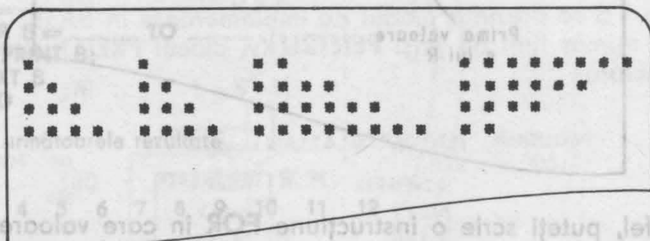
Pentru mărirea lizibilităţii programelor care conţin cicluri suprapuse se recomandă o scriere în "fierăstrău" a acestora (deplasare spre dreapta a ciclurilor interioare).

Aplicație. Scrieți un program BASIC care să listeze 4 linii de asteriscuri, fiecare linie conținînd trei asteriscuri.

```
10 FOR I=1 TO 4          RUN
20    FOR J=1 TO 3       *  *  *
30      PRINT "*";       *  *  *
40    NEXT J             *  *  *
50    PRINT              *  *  *
60 NEXT I
```

TEST

Modificați programul precedent astfel încît să tipărească următoarele rînduri:



□ Codificarea în limbajul BASIC-PRAE

vol. 2, pag. 205

Particularități ale programării

În limbajul BASIC-PRAE, spre deosebire de limbajul BASIC implementat pe calculatorul aMIC se poate vorbi de un „mariage" între instrucțiunile **PRINT** și **INPUT.** Acest aspect este ilustrat în program prin instrucțiunea:

```
240 INPUT "D="; D
```

Lansînd în execuție programul cu comanda **RUN,** pe ultimul rînd al ecranului va apare afișat:

```
D = ?
```

Punînd semnul de întrebare, calculatorul personal PRAE ne semnalează că așteaptă de la noi valoarea lui D sau altfel spus (vezi tabela de variabile) valoarea densității.

După cum precizam, calculatorul este "înarmat" cu foarte multă răbdare și așteaptă (pe timpul nostru (!)) pînă ne hotărîm să tastăm 0.7 — valoarea densității benzinei. Executînd instrucțiunea **INPUT** calculatorul va atribui valoarea 0.7 variabilei D. Execuția programului continuă cu linia 250, apoi cu linia 260 ș.a.m.d.

Formatul general al instrucţiunii **INPUT** (extinse) este:

Format general
⟨nr. linie⟩ **INPUT** [⟨constanta şir⟩;] ⟨variabila 1⟩ [, ⟨variabila 2⟩, . . .]

De notat că variabilele se separă prin virgule, iar după constanta şir (opţională) se pune simbolul " ; ".

Aplicaţie. Se consideră următorul program BASIC-PRAE care calculează volumul unui paralelipiped:

```
10 INPUT "L1, L2, L3"; L1, L2, L3
20 V=L1*L2*L3
30 PRINT "VOLUMUL=";V
40 END
```

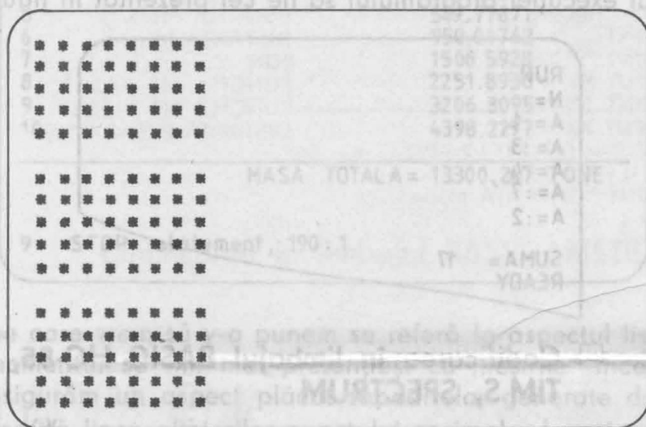
În momentul execuţiei instrucţiunii **INPUT** din linia 10, **PRAE**-ul afişează mesajul "L1, L2, L3?" iar după ce am introdus prima valoare (lungimea) va afişa încă două semne de întrebare, avertizîndu-ne în acest fel că încă-i sîntem . . . datori. Dacă, din neatenţie vom tasta ceva diferit de un număr vom fi imediat "sancţionaţi" cu mesajul "INVALID INPUT", după care se afişează din nou un semn de întrebare.

Numai după ce am răspuns corect instrucţiunii **INPUT** din linia 10, se calculează V iar apoi, pe rîndul următor "VOLUMUL=" urmat de valoarea variabilei V calculate în linia 20.

Ciclurile FOR-NEXT aparţinînd limbajului BASIC-PRAE, spre deosebire de cele din BASIC-aMIC, prezintă, în plus, următoarele **particularităţi:**

- variabila de control a ciclului poate fi modificată în interiorul buclei;
- într-o buclă **FOR-NEXT** pot fi imbricate oricîte cicluri dorim, cu condiţia ca acestea să nu se intersecteze.

Aplicaţie. Scrieţi un program BASIC-PRAE care să afişeze:



```
10 FOR I=1 TO 3
20 PRINT
30   FOR J=1 TO 5
40     FOR K=1 TO 8
50     PRINT "*";
60     NEXT K
70   PRINT
80   NEXT J
90 NEXT I
100 END
```

TESTE

1) Se consideră următorul program BASIC:

```
10 FOR I=1 TO 10 STEP 2
20
30 NEXT I
40 PRINT
50 PRINT "I="; I
60 END
```

Precizaţi rezultatul execuţiei acestui program.

2) Completaţi răspunsul dat de calculator pentru programele de mai jos:

a)
```
10 X=3
20 FOR Y=X TO 4*4 STEP X
30   PRINT Y;
40 NEXT Y
50 END
```

b)
```
10 FOR I=100 TO 10 STEP-10
20   PRINT I;
30 NEXT I
40 END
```

c)
```
10 FOR A=27 TO 18 STEP-3
20   PRINT A;
30 NEXT A
40 END
```

d)
```
10 FOR I=5 TO 7.5 STEP.25
20   PRINT I;
30 NEXT I
40 END
```

e)
```
10 FOR I=5 TO 7.5 STEP.5
20   PRINT I,
30 NEXT I
40 END
```

f)
```
10 A=0
20 C=1
30 FOR I=1 TO 4
40   A=A+1
50   C=C*I
60 NEXT I
70 PRINT "A= "; A
80 PRINT "C= "; C
90 END
```

g)
```
10 S=0
20 FOR I=1 TO 7 STEP 2
30   S=S+1
40 NEXT I
50 PRINT "S= "; S
60 END
```

3) Se consideră următorul program BASIC:

```
10 S=0
20 PRINT "N= : ";
30 INPUT N
40 FOR I=1 TO N
45   PRINT "A= :";
50   INPUT A
60   S=S+A
70 NEXT I
75 PRINT
80 PRINT "SUMA= "; S
90 END
```

Ce valoare trebuie să introduceţi dinamic pentru variabila N, astfel încît rezultatul execuţiei programului să fie cel prezentat în figură:

```
RUN
N= :
A= :1
A= :3
A= :4
A= :7
A= :2

SUMA=   17
READY
```

□ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 205

Particularităţi

Limbajul BASIC implementat pe calculatoarele personale HC-85, TIM S, SPECTRUM combină de asemenea pe **PRINT** şi **INPUT** într-o sin-

gură instrucţiune **INPUT**. Această "scurtătură" se poate vedea şi în linia 30 a programului:

30 **INPUT** "DENSITATEA BENZINEI = " ; D

La executarea instrucţiunii **INPUT** se afişează mai întîi pe terminalul utilizatorului literalul declarat în instrucţiune: "DENSITATEA BENZINEI = ", urmat de un semn de întrebare (?).

```
DENSITATEA BENZINEI =? 0.7
```

Dacă literalul a fost omis, se afişează doar "?". În continuare se aşteaptă ca utilizatorul să introducă datele solicitate, în ordinea indicată, separate prin virgule.

Cît priveşte utilizarea buclei **FOR-NEXT** în limbajul BASIC HC-85, TIM S, SPECTRUM trebuie făcută următoarea **remarcă:**

Variabila de control a buclei trebuie să aibă numele format dintr-o singură literă.

În sfîrşit, o ultimă observaţie privind forma de editare a rezultatelor:

```
RAZA                    MASA

2                       35.185838
3                       118.7522
4                       281.4867
5                       549.77871
6                       950.01762
7                       1508.5928
8                       2251.8936
9                       3206.3095
10                      4398.2297
-----------------------------------------
           MASA  TOTALA= 13300,247 TONE

9   STOP  statement, 190:1
```

Intrebarea pe care vrem să v-o punem se referă la aspectul listei cu rezultate. Este momentul să fim mai pretenţioşi cu „ieşirile" încercînd pe cît posibil să asigurăm un aspect plăcut rapoartelor generate de calculator.

Nu-i aşa că lipsa alinierilor punctului zecimal „deranjează" puţin un ochi avizat, obişnuit numai cu ordinea?

În BASIC HC-85, TIM S, SPECTRUM aceste „pretenţii" pot fi satisfăcute, nu fără o mică dificultate cu instrucţiunea **STR$** pe cînd în alte „dialecte" există căi mult mai simple (instrucţiunea **PRINT USING**).

Aplicații

Introduceți și executați următoarele programe:

a)
```
10 FOR i=1 TO 8
20   PRINT TAB 4; i; TAB 10; i ↑ 2
30 NEXT i
```

b)
```
10 LET m=3
20 LET n=14
30 FOR i=m TO n
40   PRINT TAB 4; i; TAB 6; i/10; TA
     B 14; i/m
50 NEXT i
```

c)
```
10 FOR i=100 TO 1 STEP-12.5
20   PRINT TAB 8; i
30 NEXT i
```

d)
```
10 FOR i=10 TO 1 STEP-0.175
20   PRINT TAB 10; i
30 NEXT i
```

e)
```
10 FOR i=1 TO 0
20   PRINT "*"
30 NEXT i
```

f)
```
10 FOR I=(X=6)*33 TO 31
20   PRINT CHR$ (143)
30 NEXT I
```

g)
```
 5 FOR I=1 TO 5
10   INPUT "N= "; N
20   PRINT "N= "; N
30   PRINT N; "N*N"; N ↑ 2
40 NEXT I
50 STOP
```

TEST

Precizați dacă rezultatele execuției celor două programe (a, b) sînt identice.

a)
```
10 FOR I=1 TO 6
20 PRINT "*";
30 NEXT I
40 PRINT "...."
50 END
```

b)
```
10 FOR I=1 TO 6
20   PRINT "*";
30 NEXT I
40 PRINT
50 PRINT "...."
60 END
```

Aplicație. Fiind date coordonatele mai multor perechi de puncte (acest număr se citește ca parametru) M(X1, Y1), N(X2, Y2) să se determine distanța MN.

Aplicație numerică: M(3,4); N(0,0) și M(–3.4,5.75), N(3.125,2). Schema logică a programului este prezentată în figura 4.7.

```
 10 PRINT
 20 PRINT "NUMĂR PERECHI
    PUNCTE"
 30 INPUT N
 40 FOR I=1 TO N
 45   PRINT
 50   PRINT "PUNCTUL M"
 60   INPUT X1, Y1
 70   PRINT "PUNCTUL N"
 80   INPUT X2, Y2
 90   LET D=((X2–X1) ↑ 2+(Y2–
      –Y1) ↑ 2) ↑ 0.5
100   PRINT "DISTANȚA MN="; D
110 NEXT I
120
130 END
```

```
RUN

NUMĂR PERECHI PUNCTE? 2
PUNCTUL M? 3, 4
PUNCTUL N? 0, 0
DISTANȚA MN=5
RUN
PUNCTUL M?–3.4,5.75
PUNCTUL N? 3.125,2
DISTANȚA MN=7.52583
```

TEST

Precizați care din următoarele două programe este scris corect:

a)
```
10 FOR A=1 TO 8
20 FOR B=2 TO 7
30   FOR C=2.3 TO 6.1
40   NEXT C
50 FOR D=A TO B
60 NEXT D
70 NEXT B
80 NEXT A
```

b)
```
10 FOR  A=1 TO 8
20 FOR  B=2 TO 7
30 FOR  C=2.3 TO 6.1
40 NEXT A
50 NEXT C
60 FOR  D=A TO B
70 NEXT D
80 NEXT B
```

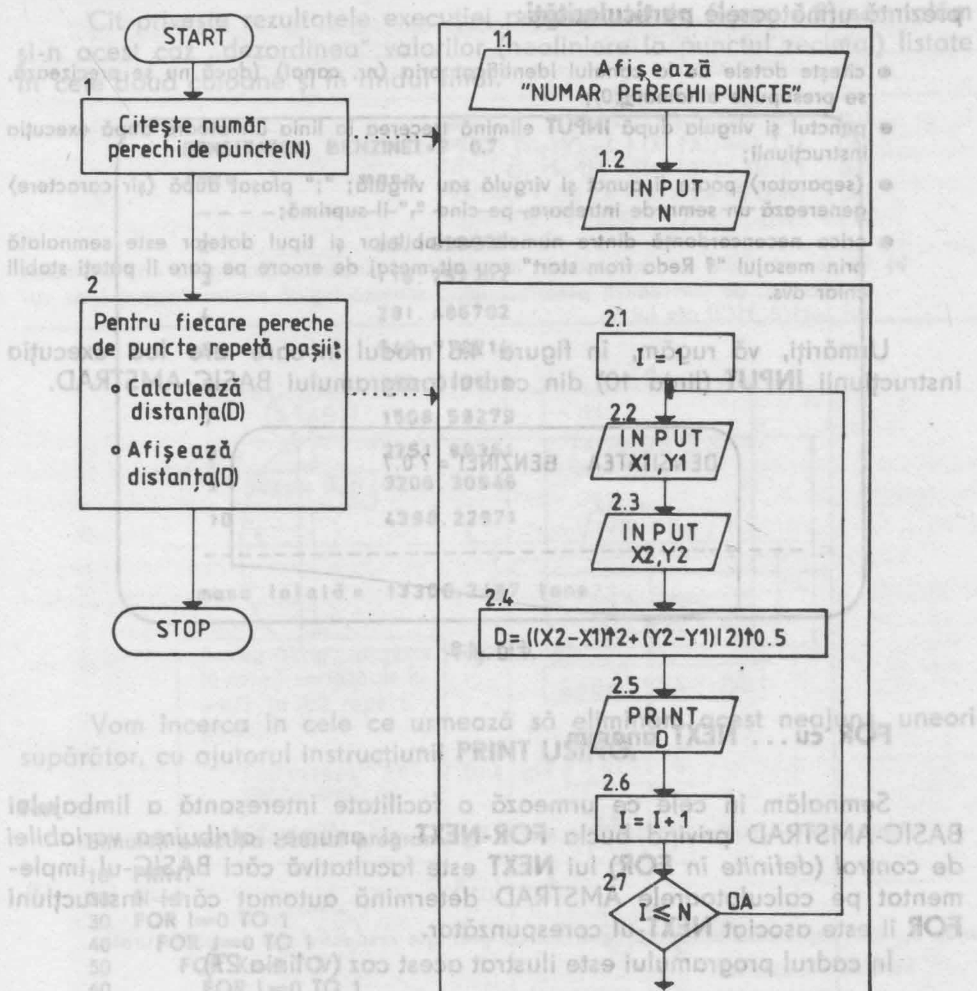


Fig. 4.7.

□ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 206

Instrucţiunea INPUT

În limbajul BASIC-AMSTRAD instrucţiunea **INPUT** al cărei format general este:

Format general
⟨nr. linie⟩ **INPUT** [#⟨nr. canal⟩,] [;] [⟨şir caractere⟩ ⟨separator⟩] ⟨listă variabile⟩

prezintă următoarele **particularități**:

- citește datele de la canalul identificat prin ⟨nr. canal⟩ (dacă nu se precizează, se presupune automat #0);
- punctul și virgula după **INPUT** elimină trecerea la linia următoare după execuția instrucțiunii;
- ⟨separator⟩ poate fi punct și virgulă sau virgulă; ";" plasat după ⟨șir caractere⟩ generează un semn de întrebare, pe cînd "," îl suprimă;
- orice neconcordanță dintre numele variabilelor și tipul datelor este semnalată prin mesajul "? Redo from start" sau alt mesaj de eroare pe care îl puteți stabili chiar dvs.

Urmăriți, vă rugăm, în figura 4.8 modul în care are loc execuția instrucțiunii **INPUT** (linia 10) din cadrul programului BASIC-AMSTRAD.

```
DENSITATEA   BENZINEI = ? 0.7
```

Fig. 4.8.

FOR cu . . . NEXT anonim

Semnalăm în cele ce urmează o facilitate interesantă a limbajului BASIC-AMSTRAD privind bucla **FOR-NEXT** și anume: *atribuirea variabilei de control (definite în* **FOR**) lui **NEXT** este facultativă căci BASIC-ul implementat pe calculatoarele AMSTRAD determină automat cărei instrucțiuni **FOR** îi este asociat **NEXT**-ul corespunzător.

În cadrul programului este ilustrat acest caz (v. linia 27).

```
25 FOR i=1 TO 40
26   PRINT "–";
27 NEXT
```

Cît privește . . . bunicul lui **FOR-NEXT** AMSTRAD vă prezentăm un program pe care l-am mai întîlnit deja în cadrul lucrării, dar de această dată scris în limbaj BASIC-AMSTRAD.

```
10 FOR I=1 TO 3
20   FOR J=1 TO 2*I
30     FOR K=1 TO 2*(J+I)
40       PRINT "*";
50 NEXT K, J, I
60 END
```

Observație. Pentru neintersectarea instrucțiunilor **FOR**, am fost nevoiți să specificăm variabilele de control ale celor trei bucle, în ordine inversă (v. linia 50).

Cît privește rezultatele execuției programului (v. figura 4.9) semnalăm și-n acest caz „dezordinea" valorilor (nealiniere la punctul zecimal) listate în cele două coloane și în rîndul final.

```
DENSITATEA BENZINEI =? 0.7
raza          masa
-----------------------------------
2             35.1858377
3             118.752202
4             281.486702
5             549.778714
6             950.017619
7             1508.59279
8             2251.89361
9             3206.30946
10            4398.22971
-----------------------------------
masa totală = 13300.2467 tone
```

Fig. 4.9.

Vom încerca în cele ce urmează să eliminăm acest neajuns, uneori supărător, cu ajutorul instrucțiunii **PRINT USING.**

Test

Simulați execuția acestui program

```
10 PRINT
20 N=1
30 FOR I=0 TO 1
40   FOR J=0 TO 1
50     FOR K=0 TO 1
60       FOR L=0 TO 1
70         FOR M=0 TO 1
80           PRINT N; I; J; K; L; M
90           N=N+1
100        NEXT M
110      NEXT L
120    NEXT K
130  NEXT J
140 NEXT I
150 END
```

Aplicații

a) Fiind date coordonatele mai multor perechi de puncte (acest număr se citește ca parametru) M(X1, Y1), N(X2, Y2) și P(X, Y) mijlocul segmentului (MN) să se determine distanța MP=NP.

```
10 PRINT
20 PRINT "NUMĂR PERECHI PUNCTE";
30 INPUT N
40 FOR I=1 TO N
50   PRINT "PUNCTUL M";
```

```
60  INPUT X1, Y1
70  PRINT "PUNCTUL N";
80  INPUT X2, Y2
90  LET X=(X1+X2)/2
100 LET Y=(Y1+Y2)/2
110 LET D=((X1-X) ↑ 2+(Y1-Y) ↑ 2) ↑ 0·5
120 PRINT "DISTANŢA MP="; D
130 NEXT I
140 END
```

b) Să se calculeze momentul static al suprafeţei haşurate din figura 4.10 în raport cu axa OZ. Se vor realiza pseudocodul şi schema logică pentru domeniul de valori [–H/2, H/2] ale lui Y.

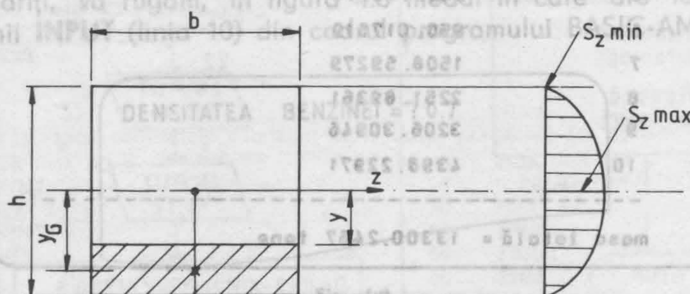


Fig. 4.10.

Pentru determinarea momentului static S_Z se utilizează relaţia de calcul:

$$S_Z = Y_G \cdot A \tag{3}$$

unde Y_G şi A reprezintă ordonata centrului de greutate respectiv aria zonei haşurate.

Se observă uşor că:

$$Y_G = y + \frac{\frac{h}{2} - y}{2} \tag{4}$$

$$A = b\left(\frac{h}{2} - y\right) \tag{5}$$

rezultînd:

$$S_Z = b\left(\frac{h}{2} - y\right)\left(y + \frac{\frac{h}{2} - y}{2}\right) = \frac{b}{2}\left(\frac{h^2}{4} - y^2\right). \tag{6}$$

Observaţie: Din relaţia (6) rezultă că S_Z variază parabolic cu y, avînd cea mai mare valoare pentru y=0 ($S_{z\,max} = \frac{bh^2}{8}$). Pentru valorile $y = \frac{h}{2}$ şi $y = -\frac{h}{2}$ valoarea momentului S_z este nulă.

În figura 4.11 (a, b) se prezintă pseudocodul şi schema logică ale aplicaţiei.

	PSEUDOCODUL
MOMENT	SEQ
	PRINT "B, H";
	INPUT B, H
A	PENTRU Y DE LA –H/2 LA H/2
	S=B * (H ↑ 2/4–Y ↑ 2)/2
	PRINT S
A	SFÎRŞIT
MOMENT	END

a)

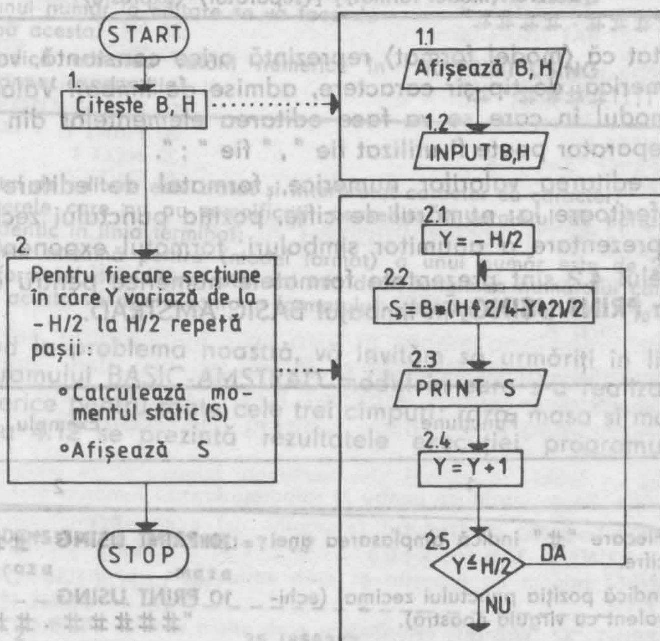


Fig. 4.11. *a)* pseudocodul; *b)* schema logică.

```
10 INPUT "B, H"; B, H
20 FOR Y=-H/2 TO H/2
30   S=B*(H*H/4-Y*Y)/2
40   PRINT "Y= "; Y; "S= "; S
50 NEXT Y
60 END
```

Editarea valorilor numerice. PRINT USING

În limbajul BASIC, noţiunea de editare trebuie înţeleasă ca pe o suprimare, inserare sau înlocuire de caractere ce aparţin unui anumit cîmp. Desigur, pot fi realizate programe BASIC şi fără utilizarea clauzelor de

editare, dar cum majoritatea programelor (BASIC) generează în mod frecvent rapoarte tipărite, iar clauzele de editare asigură o mai uşoară manipulare a sarcinilor de suprimare, inserare sau înlocuire a caracterelor, pare lipsit de sens să procedăm altfel. Se cunosc două categorii de caractere de editare: numerice şi alfanumerice. Cea mai folosită este prima categorie de caractere.

Instrucţiunea care permite utilizatorului definirea unui format propriu de editare este **PRINT USING** al cărei format general este:

Format general
PRINT [# ⟨nr. canal⟩] [⟨listă⟩] [;] [**USING** ⟨model format⟩] [⟨separator⟩ ⟨expresie⟩]

De notat că ⟨model format⟩ reprezintă orice constantă, variabilă sau expresie numerică, de tip şir caractere, admise de limbaj. Valoarea şirului reprezintă modul în care se va face editarea elementelor din parametrul ⟨listă⟩. Ca separator poate fi utilizat fie " , " fie " ; ".

Pentru editarea valorilor numerice, formatul de editare furnizează informaţii referitoare la: numărul de cifre, poziţia punctului zecimal, includerea în reprezentare a anumitor simboluri, formatul exponenţial.

În tabelul 4.2 sînt prezentate formatele numerice pentru editarea cu instrucţiunea **PRINT USING,** în limbajul BASIC AMSTRAD.

Tabelul 4.2

Indicator de cîmp	Funcţiune	Exemplu
0	1	2
#	Fiecare "#" indică amplasarea unei cifre.	10 **PRINT USING** "####", A
.	Indică poziţia punctului zecimal (echivalent cu virgula noastră).	10 **PRINT USING** "#####.##", A
,	(Rezervă o poziţie). Acest simbol neputînd figura decît imediat înaintea punctului zecimal, ne precizează că cifrele situate la stînga punctului zecimal vor fi dispuse în grupe de trei separate între ele printr-o virgulă.	10 **PRINT USING** "######,.##"; A
££	(Rezervă două poziţii). Indică plasarea acestuia imediat înaintea primei cifre sau a punctului zecimal, altfel spus, pe unul din spaţiile rezervate cifrelor.	10 **PRINT USING** "££######,.##"; A
**	(Rezervă două poziţii). Indică completarea cîmpurilor caracter terminal neocupate, existente la stînga punctului zecimal cu asteriscuri.	10 **PRINT USING** "**######,.##", A
**£	(Rezervă trei poziţii). Combină opţiunile ** şi ££.	10 **PRINT USING** "**£######,.##", A

0	1	2
$$	(Rezervă două poziţii). Indică plasarea acestuia înaintea primei cifre sau a punctului zecimal.	10 PRINT USING "$$######,.##", A
**$	(Rezervă trei poziţii). Combină opţiunile ** şi $.	10 PRINT USING "**$######,.##", A
+	Indică faptul că se doreşte să se figureze semnul numărului. Acest semn va apare înaintea numărului dacă "+" este situat la început de model format şi după număr dacă el este plasat la urmă.	10 PRINT USING "####.##", A
–	Indică faptul că plasarea semnului unui număr la editare se va face după acesta.	10 PRINT USING "####.####–," A
↑↑↑↑	Indică editarea valorii numerice în format exponenţial.	10 PRINT USING "#.###↑↑↑↑", A

Remarci

- Formatul de editare este tratat şi interpretat caracter cu caracter;
- Caracterele care nu au semnificaţii deosebite în formatul de editare sînt introduse identic în linia terminal;
- Lungimea maximă pentru ⟨model format⟩ a unui număr este de 20 caractere;
- Dacă formatul de editare este mai mic decît lungimea numărului care se doreşte extras, acesta va apare conform formatului precedat de simbolul "%".

Revenind la problema noastră, vă invităm să urmăriţi în liniile 80 şi 100 ale programului BASIC-AMSTRAD modul în care s-a realizat editarea valorilor numerice pentru toate cele trei cîmpuri: raza, masa şi masa totală.

În figura 4.12 se prezintă rezultatele execuţiei programului (noua versiune).

```
DENSITATEA BENZINEI =? 0.7
raza          masa
- - - - - - - - - - - - - - - - - - - - - - -
2             35.1858377
3             118.752202
4             281.486702
5             549.778714
6             950.017619
7             1508.59279
8             2251.89361
9             3206.30946
10            4398.22971
- - - - - - - - - - - - - - - - - - - - - - -
masa  totala =  13300.2467 tone
```

Fig. 4.12.

Încercaţi fără ajutorul nostru, să înţelegeţi bine semnificaţia formatelor de editare confruntîndu-le direct cu rezultatele obţinute.

TEST

Precizaţi rezultatul execuţiei următoarelor programe:

a)
```
5 B=3
10 FOR A=-3 TO B STEP 3
20     PRINT A
30 NEXT A
40 END
```

b)
```
5 A=4 : B=-5 : C=-2
10 FOR I=A TO B STEP C
20         PRINT I;
30 NEXT I
40 END
```

c)
```
10 S=0
20 FOR I=1 TO 100
30      S=S+I
40      I=I+1
50 NEXT I
60 PRINT "S= "; S
70 END
```

d)
```
10 P=1 : S=0
20 FOR I=5 TO 23
30      S=S+I
40      P=P*I
50 NEXT I
60 PRINT "P/S"; P/S
70 END
```

e)
```
5 INPUT "A="; A
10 FOR I=1 TO A STEP 2
20     PRINT I;
30 NEXT I
40 END
```

f)
```
5 INPUT "A, B, C="; A, B, C
10 FOR I=A TO B STEP C
20 PRINT I;
30 NEXT I
40 END
```

Observaţie. Se vor introduce dinamic valorile 8, –4, –5 corespunzătoare variabilelor A, B respectiv C.

g)
```
5 INPUT "C="; C
10 FOR A=-2 TO -11 STEP C
20    PRINT A,
30 NEXT A
40 END
```

Observaţie. Se va introduce pentru C valoarea 4.

h)
```
10 INPUT "N="; N
20 FOR I=1 TO N
30    PRINT "="
40 NEXT I
50 END
```

Observaţie. Se va introduce pentru N valoarea 4.

i)
```
10 FOR I=33 TO 63 STEP 10
20     PRINT X; TAB 10; X ↑ 0.5
30 NEXT I
40 END
```

j)
```
10 PRINT "X"; TAB (8); "Y"; TAB
   (15); "X+Y"
20 PRINT
30 FOR I=1 TO 5
40      INPUT "X, Y"; X, Y
50      PRINT X; TAB (8); Y; TAB
        (15); X+Y
60 NEXT I
70 END
```

k)
```
10 FOR N=1 TO 10
20      PRINT N, N ↑ (-2)
30 NEXT N
40 STOP
```

l)
```
10 FOR I=1 TO 10
20      N=1
30      S=I * I
40      C=I ↑ 3
50      PRINT N; TAB (5); S; TAB
        (12); C
60 NEXT I
70 END
```

m)
```
10 FOR I=1 TO 10
20      PRINT I; TAB (5); I ↑ 2;
        TAB (12); I ↑3
30 NEXT I
40 END
```

n)
```
10 FOR I=1 TO 10
20    FOR J=1 TO N
30       PRINT "A";
40    NEXT J
50 PRINT
60 NEXT I
```

o)
```
10 A=1
20 FOR I=1 TO 5
30    A=2*A
40    FOR J=1 TO A
50       PRINT "C";
60    NEXT J
70 PRINT
80 NEXT I
90 STOP
```

p)
```
10 FOR I=1 TO 21 STEP 2
20    A=I*I
30    B=A*I
40    PRINT I;
50    PRINT "        "; A;
60    PRINT "        "; B
70 NEXT I
80 STOP
```

r)
```
10 FOR I=5 TO 7
20    FOR K=1 TO 10
25       P=K*I
30       PRINT K; "X"; I; "="; P
40    NEXT K
50    PRINT
60 NEXT I
```

s)
```
10 INPUT "LUNGIMEA="; L
20 INPUT "LĂȚIMEA="; B
30 FOR I=1 TO L
40    FOR J=1 TO B
50       PRINT "X";
60    NEXT J
70 PRINT
80 NEXT I
90 STOP
```

Observație. Pentru L și B se vor introduce valorile 4 și 9, (vezi s).

t)
```
10 FOR I=1 TO 32
20    PRINT "-";
30 NEXT I
40 STOP
```

u)
```
10 FOR I=1 TO 32
20    PRINT "-";
30 NEXT I
35 PRINT
40 LET S=14
50 PRINT TAB (S); "<=>"
55 PRINT
60 FOR I=1 TO 32
70    PRINT "-";
80 NEXT I
```

□ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 207

La scrierea programului s-au avut în vedere particularitățile de programare privind instrucțiunile **INPUT** și **FOR-NEXT**. În limbajul BASIC-COMMODORE instrucțiunea **INPUT** (al cărei format general este identic cu cel de la BASIC-AMSTRAD) prezintă următoarele **particularități:**

- citește datele de la canalul identificat prin ⟨nr. canal⟩ (dacă nu se precizează se consideră automat tastatura);
- dacă instrucțiunea conține ⟨șir caractere⟩ acesta este afișat înaintea cererii de date, urmat de "?";
- dacă se răspunde cu mai puține date se afișează pe monitor două semne de întrebare;
- dacă se răspunde cu mai multe date se tipărește mesajul:
  ? EXTRA IGNORED
- dacă se răspunde cu date de tip necorespunzător se afișează mesajul:
  ? REDO FROM START

În cadrul programului instrucțiunea **INPUT** ocupă linia 10.

În ceea ce privește utilizarea buclei **FOR-NEXT** menționăm că limbajul BASIC-COMMODORE admite următoarele facilități:

- dacă pe **NEXT** nu este specificat nici un nume de variabilă, el corespunde primului **FOR** care îl precede;
- pot fi imbricate mai multe cicluri **FOR-NEXT** cu condiția ca acestea să nu se intersecteze iar variabilele de control corespunzătoare să fie unice.

În program bucla **FOR-NEXT** ocupă liniile 40–85.

Observație. Limbajul BASIC-COMMODORE nu admite facilități **(PRINT-USING)** privind editarea valorilor numerice și nenumerice.

TEST

Precizați valoarea lui S ("S=") în urma execuției programului:

```
10 S=0                        RUN
20 FOR I=1 TO 4               A, B ? 1, 2
30   INPUT "A, B"; A, B       A, B ? 2, 3
40   S=S+A * B                A, B ? 3, 4
50 NEXT I                     A, B ? 4, 5
60 PRINT                      S=
70 PRINT "S="; S
80 END
```

□ Particularități ale programării în limbajul BASIC-80

vol. 2, pag. 207

Instrucțiunea INPUT

În limbajul BASIC-80, instrucțiunea **INPUT**, al cărei format general este:

Format general
⟨nr. linie⟩ **INPUT** [;] [⟨șir caractere⟩ ;] ⟨listă variabile⟩

prezintă următoarele **particularități:**

- dacă instrucțiunea conține ⟨șir caractere⟩, acesta este afișat înaintea cererii de date, urmat de "?";
- pentru a suprima caracterul "?" se poate folosi, " , " după ⟨șir caractere⟩ sau " ; " după **INPUT**;
- datele introduse se atribuie variabilelor din ⟨listă variabile⟩ în ordine;
- dacă se răspunde cu mai multe sau mai puține date sau de tip necorespunzător se afișează mesajul:

? Redo from start.

În cadrul programului instrucțiunea **INPUT** ocupă linia 60.

Instrucțiunea FOR

În limbajul BASIC-80, bucla **FOR-NEXT** prezintă următoarele **particularități:**

- ciclurile **FOR-NEXT** pot fi suprapuse (imbricate) dar trebuie să nu se intersecteze și să aibă variabile de control diferite;
- atribuirea variabilei de control lui **NEXT** este facultativă;
- o instrucțiune **NEXT** fără variabilă este asociată cu ultima instrucțiune **FOR**.

În program instrucțiunea **FOR-NEXT** ocupă liniile 130, 140, 150, 160, 170, 180.

Odată cu parcurgerea instrucţiunilor cuprinse în ciclul **FOR-NEXT** al programului aţi sesizat ceva deosebit la instrucţiunea din linia 140?

```
130 FOR R=2 TO 10
140   V=2 * PI *R ↑ 3
.
.
.
170   PRINT R, M
180 NEXT R
```

Cei mai grăbiţi dintre dumneavoastră n-au remarcat poate faptul că valoarea lui PI a fost definită în linia 115 într-un mod cu totul nou.

115 PI = 4 * **ATN** (1) : **REM** "Numărul PI"

Nu vă alarmaţi, pentru că nu este nimic complicat.

Amintiţi-vă din liceu că

$$\operatorname{arctg} 1 = \frac{\pi}{4} \qquad (8)$$

sau:

$$\pi = 4 \cdot \operatorname{arctg} 1 \qquad (9)$$

În BASIC (80) relaţia (9) se codifică exact ca în linia 115 cu menţiunea că **ATN**(1) reprezintă funcţia aritmetică arctangentă de 1.

Menţionăm, de asemenea, faptul că în cadrul programului nu ne-am preocupat de editarea cu format (v. linia 170) a valorilor numerice (v. fig. 4.13), ceea ce nu înseamnă că dumneavoastră nu puteţi realiza acest lucru singuri, ca o aplicaţie privind utilizarea instrucţiunii **PRINT USING** în limbajul BASIC-80. De notat că formatele numerice de editare prevăd pe lîngă indicatorii de cîmp: "#", "+", "−", "**", "$$", "**$", " , ", "∧∧∧∧" întîlniţi la limbajul BASIC-AMSTRAD (v. tabelul 4.2) şi indicatorul "\" cu semnificaţia că următorul caracter din format se afişează.

```
================================
Raza             Masa
================================
 2               35.1858
 3               118.752
 4               281.487
 5               549.779
 6               950.018
 7               1508.59
 8               2251.89
 9               3206.31
 10              4398.23
================================
Masa totală =    13300.2    tone
```

Fig. 4.13.

Remarcă. Dacă numărul de cifre specificat este mai mare decît 24 se afişează un mesaj de eroare.

TEST

Simulaţi execuţia acestui program:

```
10  FOR I=1 TO 3
20    FOR J=1 TO 3
30      FOR K=1 TO 3
40        PRINT I; J; K
50      NEXT K
60    NEXT J
70  NEXT I
80  END
```

în maniera:

Instrucţiune	I	J	K
	Valorile variabilelor I, J, K		
40 PRINT I; J; K	1	1	1
	1	1	2
	1	1	3
	1	2	1
	1	2	2
	1	2	3
	1	3	1
	1	3	2
	1	3	3
	.	.	.
	.	.	.
	.	.	.
40 PRINT I; J; K	3	3	3

Aplicaţii

a) Să se editeze următorul triunghi al numerelor:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

```
10 FOR I=1 TO 10
20   FOR J=1 TO I
30     PRINT J;
40   NEXT J
50   PRINT
60 NEXT I
70 END
```

b) Determinaţi al n-lea termen dintr-un şir Fibonacci. Se vor realiza tabela de variabile, schema logică şi programul BASIC.

Un şir Fibonacci este un şir de numere întregi în care fiecare număr este egal cu suma celor două numere precedente. Altfel spus, suma primului şi celui de-al doilea număr este egală cu cel de-al treilea număr, suma celui de-al doilea număr şi celui de-al treilea este egală cu al patrulea număr etc.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

0+1 1+1 1+2 2+3

Se va utiliza formula de recurenţă

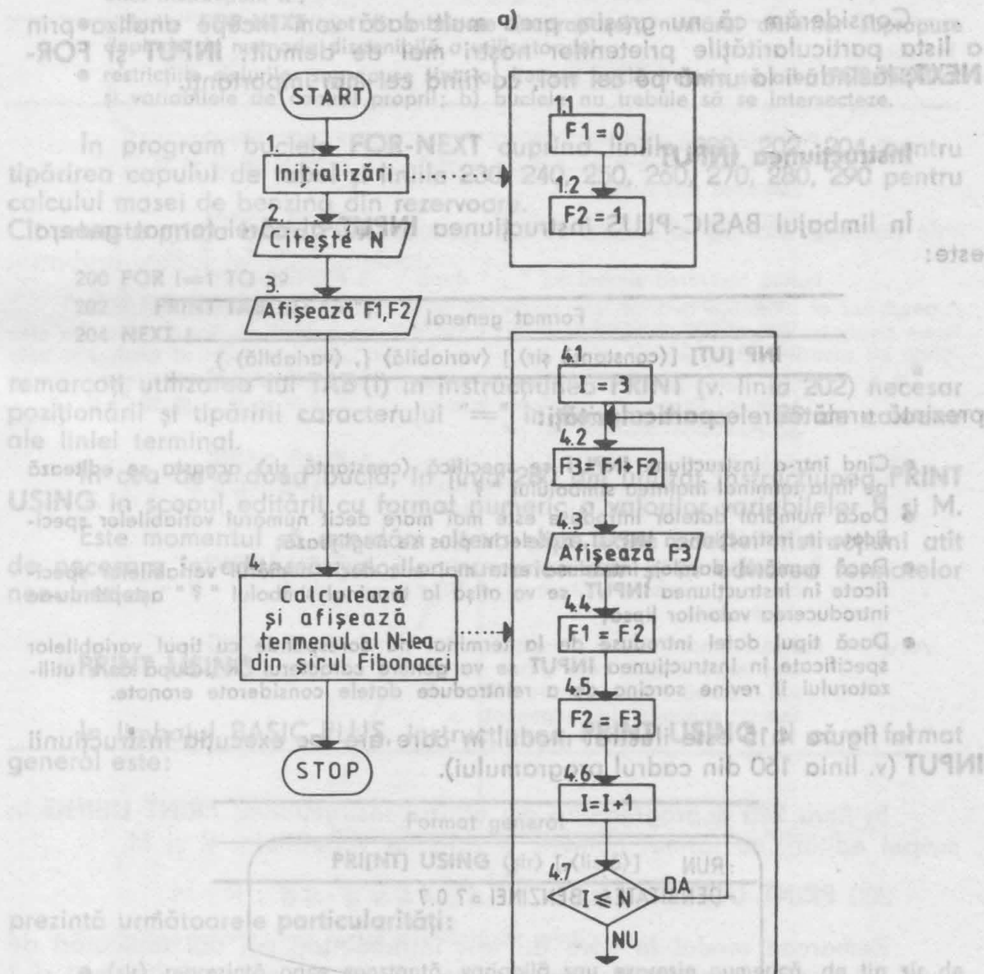
$$F_N = F_{N-1} + F_{N-2},\ N > 2,\ F_1 = 0,\ F_2 = 1 \quad (10)$$

În figura 4.14 (a, b) sînt ilustrate tabela de variabile și schema logică ale aplicației.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
N : rangul termenului căutat	I : variabila de control a structurii de iterație PENTRU	
	F1 : primul număr Fibonacci	F3 : termenul căutat din șirul Fibonacci
	F2 : al doilea număr Fibonacci	

a)



b)

Fig. 4.14. *a)* tabela de variabile; *b)* schema logică.

□ Particularități ale programării în limbajul BASIC-PLUS

vol. 2, pag. 208

Programul pe care-l analizăm acum găzduiește pe lîngă cunoștințele noastre mai vechi: **REM** (v. liniile 100–150, 170, 220–270, 290, 300, 370), cei doi amici nedespărțiți **FOR** și **NEXT** (v. liniile 200, 204, 240, 290), **INPUT** (v. linia 160), numeroșii **PRINT, PRINT USING** (v. linia 280) și invitați de „onoare", cum sînt: modificatorul **FOR** (v. liniile 310, 360, 370), variabila alfanumerică A$ (v. linia 300) ș.a.

Considerăm că nu greșim prea mult dacă vom începe analiza prin a lista particularitățile prietenilor noștri mai de demult: **INPUT** și **FOR-NEXT**; lăsîndu-i la urmă pe cei noi, ca fiind cei mai importanți.

Instrucțiunea INPUT

În limbajul BASIC-PLUS instrucțiunea **INPUT** al cărei format general este:

Format general
INP [UT] [⟨constantă șir⟩,] ⟨variabilă⟩ {, ⟨variabilă⟩ }

prezintă următoarele **particularități:**

- Cînd într-o instrucțiune **INPUT** se specifică ⟨constantă șir⟩ aceasta se editează pe linia terminal înaintea simbolului " ? ";
- Dacă numărul datelor introduse este mai mare decît numărul variabilelor specificate în instrucțiunea **INPUT,** datele în plus se neglijează;
- Dacă numărul datelor introduse este mai mic decît numărul variabilelor specificate în instrucțiunea **INPUT,** se va afișa la terminal simbolul " ? " așteptîndu-se introducerea valorilor lipsă;
- Dacă tipul datei introduse de la terminal nu corespunde cu tipul variabilelor specificate în instrucțiunea **INPUT** se va genera caracterul "R", după care utilizatorului îi revine sarcina de a reintroduce datele considerate eronate.

În figura 4.15 este ilustrat modul în care are loc execuția instrucțiunii **INPUT** (v. linia 160 din cadrul programului).

```
:RUN
-DENSITATEA BENZINEI =? 0.7
.
```

Fig. 4.15.

Bucla FOR-NEXT. Particularităţi

În limbajul BASIC-PLUS instrucţiunile **FOR... TO** şi **NEXT** prezintă următoarele **particularităţi:**

- variabila de control a ciclului poate fi orice variabilă numerică simplă de tip întreg sau real;
- variabila de control poate fi modificată în interiorul ciclului;
- este posibil transferul controlului în exteriorul buclei înainte de terminarea execuţiei buclei prin intermediul instrucţiunilor **GO TO, ON ... GOTO, IF;**
- controlul poate fi transferat în interiorul unui ciclu, dar acesta se va manifesta diferit, funcţie de realizarea sau nu a iniţializării ciclului;
- **FOR-NEXT** nu poate să apară pe o linie imediată sau ca parte componentă a unei instrucţiuni **IF**;
- ciclurile **FOR-NEXT** pot fi imbricate (suprapuse); numărul ciclurilor suprapuse depinde de memoria disponibilă a utilizatorului;
- restricţiile ciclurilor suprapuse sînt: a) fiecare buclă trebuie să aibă **FOR-NEXT**-ul şi variabilele de control proprii; b) buclele nu trebuie să se intersecteze.

În program buclele **FOR-NEXT** cuprind liniile 200, 202, 204 pentru tipărirea capului de tabel şi liniile 230, 240, 250, 260, 270, 280, 290 pentru calculul masei de benzină din rezervoare.
Cît priveşte prima buclă **FOR**

```
200 FOR I=1 TO 29
202     PRINT TAB (I); "=";
204 NEXT I
```

remarcaţi utilizarea lui **TAB** (I) în instrucţiunea **PRINT** (v. linia 202) necesar poziţionării şi tipăririi caracterului "=" în fiecare din cele 29 de coloane ale liniei terminal.

În cea de-a doua buclă, în linia 280 am utilizat instrucţiunea **PRINT USING** în scopul editării cu format numeric a valorilor variabilelor R şi M.

Este momentul să acordăm cîteva clipe bune acestei instrucţiuni atît de necesare în editarea valorilor numerice dar şi în editarea formatelor nenumerice.

PRINT USING

În limbajul BASIC-PLUS, instrucţiunea **PRINT USING** al cărei format general este:

Format general
PRI[NT] USING ⟨şir⟩ [,⟨listă⟩]

prezintă următoarele **particularităţi:**

- ⟨şir⟩ reprezintă orice constantă, variabilă sau expresie numerică, de tip şir de caractere, admisă de limbaj;
- ⟨lista⟩ reprezintă o listă de elemente (admise de limbaj) separate între ele prin caracterele " " sau " ; ";
- ⟨lista⟩ se poate termina prin caracterul " , ", " ; " sau spaţiu;

- elementele din ⟨listă⟩ sînt parcurse în timpul execuţiei programului, paralel cu caracterele din ⟨şir⟩;
- separatorii dintre elementele listei nu au efect asupra formatului de editare;
- dacă după ultimul element al listei urmează caracterul " , " sau " ; ", următoarea instrucţiune **PRINT USING** sau **PRINT** va tipări pe aceeaşi linie terminal;
- formatul de editare este tratat şi interpretat caracter cu caracter;
- caracterele care nu au semnificaţii deosebite în formatul de editare sînt introduse identic în linia terminal.

În tabelul 4.3 sînt prezentate formatele numerice pentru editarea cu instrucţiunea **PRINT USING** în limbajul BASIC-PLUS a valorilor numerice.

Tabelul 4.3

Indicator de cîmp	Funcţiune	Exemplu
#	Precizează că un cîmp caracter din linia terminal va fi ocupat de o cifră zecimală sau spaţiu.	30 **PRINT USING** " # # # # # # ", C
·	Indică poziţia punctului zecimal.	5 **PRINT USING** " # # # · # # ", C
,	Indică inserarea simbolului " , " după fiecare grup de trei cifre, pornind spre stînga punctului zecimal, dacă acesta există.	5 **PRINT USING** " ####·#### ", A
**	Indică completarea cîmpurilor caracter terminal neocupate, existente la stînga punctului zecimal, cu " * ".	5 **PRINT USING** " **# # · # # ", B
$ $	Indică plasarea simbolului " $ " înaintea numărului editat şi eventuala completare cu spaţiu în stînga punctului zecimal.	5 **PRINT USING** " $$# · ### "; A
–	Precizează faptul că plasarea semnului unui număr la editare se va face după acesta.	5 **PRINT USING** " # # # · # – ", A
^^^^	Indică editarea valorii numerice în format exponenţial, cele patru simboluri rezervă spaţiu pentru: litera E, semnul exponentului, exponent.	5 **PRINT USING** " ###·#^^^^ ", A

În linia 280 a programului am utilizat instrucţiunea **PRINT USING** în scopul editării cu format numeric a valorilor variabilelor R şi M.

```
280 PRINT USING "##          ######·##", R ; M
```

Remarcaţi modul în care au fost întrebuinţaţi cei doi indicatori de cîmp "#" şi ".". Aţi înţeles de ce pentru R=10 se afişează 0 sub 9 şi 1 imediat la stînga? Deoarece numărul caracterelor "#" este mai mare decît numărul de cifre ale valorilor (2, 3, 4, 5, 6, 7, 8, 9) de editat, cîmpurile caracter neocupate se completează cu spaţiu (v. figura 4.16).

```
1                                                     
↓    primul                al zecelea                 
     blanc                 blanc                      
# #ƀ ƀ                     ƀ ƀ # # # # # # . # #       şablon
ƀ 2                        ƀ ƀ ƀ ƀ ƀ ƀ 3 5 . 1 9       pentru
ƀ 3                        ƀ ƀ ƀ ƀ ƀ 1 1 8 . 7 5       formatul
ƀ 4                        ƀ ƀ ƀ ƀ ƀ 2 8 1 . 4 9       de editare
ƀ 5                        ƀ ƀ ƀ ƀ ƀ 5 4 9 . 7 8
ƀ 6                        ƀ ƀ ƀ ƀ ƀ 9 5 0 . 0 2
ƀ 7                        ƀ ƀ ƀ ƀ 1 5 0 8 . 5 9
ƀ 8                        ƀ ƀ ƀ ƀ 2 2 5 1 . 8 9
ƀ 9                        ƀ ƀ ƀ ƀ 3 2 0 6 . 3 1
1 0                        ƀ ƀ ƀ ƀ 4 3 9 8 . 2 3
 :                                :
```

Fig. 4.16.

Cît priveşte spaţiile (blancurile) din formatul de editare acestea sînt introduse identic în linia terminal.

Aplicaţie. Deasupra centrului unei mese rotunde de 2 m diametru la înălţimea h este atîrnată o sursă de lumină de 100 cd. Să se determine iluminarea la marginea mesei cînd h variază în intervalul $0{,}5 \leqslant h \leqslant 0{,}9$ m (v. fig. 4.17). Se vor alcătui tabela de variabile şi schema logică.

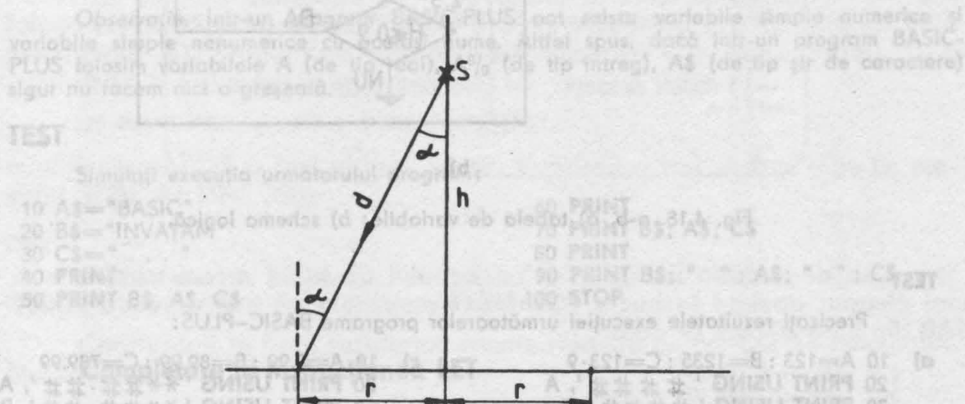


Fig. 4.17.

Iluminarea la marginea mesei este dată de relaţia

$$E = \frac{I}{d^2}\cos\alpha = \frac{I}{r^2+h^2}\cdot\frac{h}{\sqrt{r^2+h^2}} = \frac{Ih}{(r^2+h^2)^{3/2}} \quad (11)$$

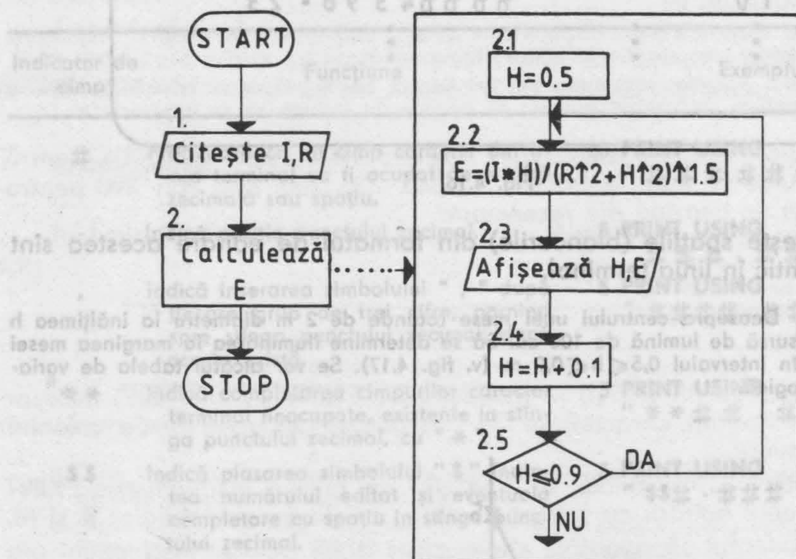
unde I este intensitatea luminoasă a sursei, E iluminarea, h înălţimea la care este situată sursa de lumină, r=D/2 (D este diametrul mesei).

Tabela de variabile și schema logică sînt prezentate în figura 4.18 (a, b).

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
R : Raza mesei I : intensitatea luminoasă a sursei	H : înălțimea	E : iluminarea la marginea mesei

a)



b)

Fig. 4.18. a-b. *a*) tabela de variabile; *b*) schema logică.

TEST

Precizați rezultatele execuției următoarelor programe BASIC–PLUS:

a)
```
10 A=123 : B=1235 : C=123·9
20 PRINT USING '####', A
30 PRINT USING '####', B
40 PRINT USING '###', C
50 END
```

b)
```
10 A=304.23 : B=25.77 : C=
   25.779 : D=111.99
20 PRINT USING '###.##', A
30 PRINT USING '###.##', B
40 PRINT USING '###.##', C
50 PRINT USING '##.##', D
60 END
```

c)
```
10 A=8.99 : B=89.99 : C=789.99
20 PRINT USING '**##.##', A
30 PRINT USING '**##.##', B
40 PRINT USING'**##.##', C
50 END
```

d)
```
10 A=5.842 : B=17.38 : C=17.385
20 PRINT USING '$$#.##', A
30 PRINT USING '$$#.##', B
40 PRINT USING '$$#.##', C.
50 END
```

e)
```
10 A=-2.3 : B=3.5
20 PRINT USING '##.#', A
30 PRINT USING '##.#', B
40 END
```

f)
```
10 A=835 : B=12345.67
20 PRINT USING '##,###', A
30 PRINT USING '$$##,
   ##.#', B
40 END
```

g)
```
10 A=2 : B=22 : C=222 :
   D=222222 : E=22.22 : F=22.222
20 PRINT USING '###.# ^^^^',  A
30 PRINT USING '###.# ^^^^',  B
40 PRINT USING '###.# ^^^^',  C
50 PRINT USING '###.# ^^^^',  D
60 PRINT USING '###.# ^^^',   E
70 PRINT USING '###.# ^^^^^', F
80 END
```

Constante şi variabile alfanumerice

A sosit şi rîndul „invitaţilor"! Să facem cunoştinţă cu variabila alfanumerică A$ (v. linia 300).

```
300 A$="="! INIT. VARIABILA ALFA
```

Puţină teorie! Limbajul BASIC permite manipularea nu numai a informaţiilor numerice ci şi a celor nenumerice, recunoscute sub forma şirurilor de caractere (o succesiune de litere, cifre sau caractere speciale nu mai multe de 255).

Constanta şir de caractere este formată dintr-un şir de caractere inclus între simbolurile " ' " sau " " ". Cît priveşte *variabilele simple nenumerice* acestea sînt folosite pentru definirea, identificarea sau referirea informaţiei nenumerice de tip şir de caractere. Tipul nenumeric al unei variabile şir de caractere se indică prin simbolul "$" adăugat numelui acesteia. Aşadar, A$ – noua noastră cunoştinţă din linia 300 este o variabilă simplă nenumerică (variabila şir de caractere) căreia i s-a atribuit constanta şir de caractere "=".

Observaţie. Într-un program BASIC-PLUS pot exista variabile simple numerice şi variabile simple nenumerice cu acelaşi nume. Altfel spus, dacă într-un program BASIC-PLUS folosim variabilele A (de tip real), A% (de tip întreg), A$ (de tip şir de caractere) sigur nu facem nici o greşeală.

TEST

Simulaţi execuţia următorului program:

```
10 A$="BASIC"
20 B$="ÎNVĂŢĂM"
30 C$=" . . . "
40 PRINT
50 PRINT B$, A$, C$
60 PRINT
70 PRINT B$; A$; C$
80 PRINT
90 PRINT B$; " "; A$; " "; C$
100 STOP
```

Completare la instrucţiunea LET

Instrucţiunea de atribuire **LET** afectează (atribuie) o valoare numerică sau un şir de caractere unei variabile simple numerice sau nenumerice (alfanumerice) sau unei variabile indexate.

Formatul general al instrucţiunii este

Format general
[LET] < variabilă > { , < variabilă > } = < expresie >

De notat că variabila din stînga semnului egal și expresia din dreapta trebuie să fie de același tip (numeric sau nenumeric). Și încă un amănunt de loc de neglijat: instrucțiunea **LET** permite în limbajul BASIC-PLUS o atribuire multiplă.

Remarcă. Instrucțiunea de atribuire din linia 300 poate fi folosită în toate versiunile BASIC din această lucrare.

Patru metode de generare a unui rînd de subliniere

Pentru generarea unui rînd cu 29 de simboluri "=" folosit în sublinierile din lista cu rezultate s-au folosit patru metode (v. figura 4.19) — pentru fiecare rînd de subliniere cîte o metodă. Astfel, pentru primul rînd necesar trasării liniei de început a capului de tabel s-a utilizat instrucțiunea

180 **PRINT** "=========="

Metoda este incomodă din motive evidente.

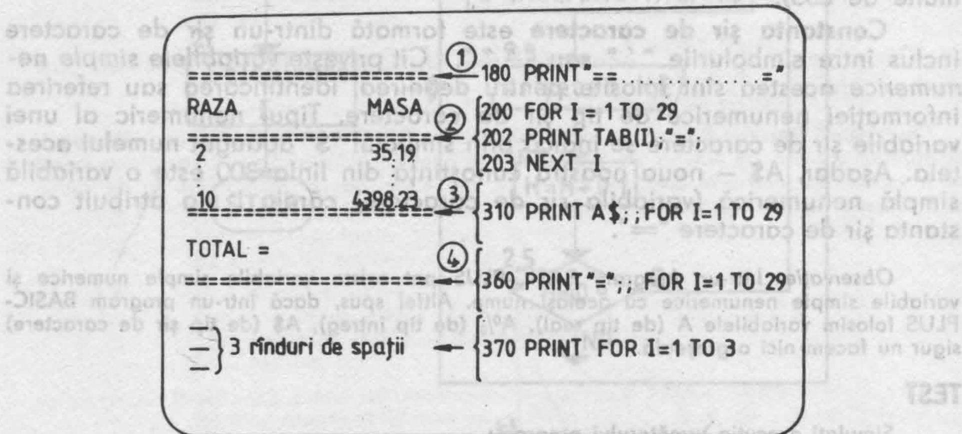


Fig. 4.19.

Cea de-a doua linie cu "=" a capului de tabel a fost trasată mult mai elegant apelînd la bucla **FOR-NEXT** care iterează instrucțiunea **PRINT TAB** din linia 202 de 29 de ori.

```
200 FOR I=1 TO 29
202     PRINT TAB (I); "=";
203 NEXT I
```

Observație. Remarcați prezența separatorului "; " la sfîrșitul instrucțiunii **PRINT** necesar tipăririi pe același rînd a caracterului "=".

Ultimele două linii ale listei au fost generate cu două instrucțiuni (v. liniile 310 și 360) care la prima vedere, de ce să nu recunoaștem, par a fi puțin ciudate! Este vorba de **modificatorul FOR.** De ce modificatorul **FOR**? vă veți întreba pe bună dreptate.

Modificatorul FOR

Desigur că există un răspuns şi la această întrebare. Utilizarea modificatorilor de instrucţiuni permite o mai bună flexibilitate în exprimarea unei condiţii. Limbajul BASIC-PLUS cunoaşte cinci modificatori: **IF, UNLESS, WHILE, UNTIL, FOR.**

Modificatorul **FOR** (cunoştinţa noastră!) creează o buclă de o instrucţiune avînd o variabilă de control indicată.

Formatul general al modificatorului **FOR** este:

Format general
< nr. linie > < instrucţiune > **FOR** < variabilă > = < exp. 1 > **TO** < exp. 2 > [**STEP** < exp. 3 >]

Remarci

- Parametrii instrucţiunii sînt aceeaşi cu cei de la instrucţiunea **FOR.**
- Modificatorul **FOR** se aplică numai instrucţiunii care îl precede.
- O instrucţiune poate fi însoţită de unul sau mai mulţi modificatori, acţiunea acestora fiind tratată de la dreapta la stînga.
- Pe o linie program multiplă modificatorul acţionează numai asupra instrucţiunii care îl precede.

Revenind la instrucţiunile din liniile 310 şi 360

```
310 PRINT A$ ; ; FOR I=1 TO 29
.
.
360 PRINT "=" ; ; FOR I=1 TO 29
```

remarcaţi cum modificatorul **FOR** este folosit pentru însoţirea instrucţiunii **PRINT** cu singura deosebire că în linia 310 în locul caracterului "=" s-a folosit variabila şir de caractere A$ a cărei valoare a fost stabilită în linia precedentă 300 (iniţializare variabilă alfanumerică – v. comentariul).

În linia 370 a aceluiaşi program

```
370 PRINT FOR I=1 TO 3! SARE 3 RINDURI
```

modificatorul **FOR** a fost utilizat pentru tipărirea a trei rînduri albe (v. mesajul STOP AT LINE 380).

TEST

Simulaţi execuţia programului cu creionul şi hîrtia. Veţi întîmpina mici dificultăţi în special la editarea cu format a valorilor numerice.

Confruntaţi în final rezultatul dumneavoastră cu cel dat de calculator. Nu vă grăbiţi!

☐ Particularităţi ale programării în limbajul ABASIC

vol. 2, pag. 209

La scrierea programului s-au avut în vedere particularităţile de programare privind instrucţiunile **INPUT, FOR-NEXT** şi **PRINT USING.**

În limbajul ABASIC, instrucțiunea **INPUT** prezintă următoarele particularități:

- dacă nu se specifică ⟨listă variabile⟩, ci numai ⟨șir caractere⟩ se face numai afișarea acestuia (șir caractere);
- numărul de valori separate prin " , " trebuie să fie cel puțin egal cu numărul de variabile din listă (valorile suplimentare se ignoră).

În cadrul programului instrucțiunea **INPUT** ocupă linia 160.

Instrucțiunea **FOR-NEXT** prezintă **particularitățile:**

- dacă pe **NEXT** nu este specificat nici un nume de variabilă, el corespunde primului **FOR** care îl precede;
- se pot imbrica mai multe cicluri **FOR-NEXT** cu terminarea pe un **NEXT** comun, sau pe **NEXT**-uri diferite;
- ciclurile imbricate nu trebuie să se intersecteze iar variabilele de control corespunzătoare trebuie să fie unice;
- **FOR-NEXT** acceptă opțiunea **WHILE,** condiția testîndu-se după testul variabilei de control.

În program bucla **FOR-NEXT** ocupă liniile 240–290.

Cît privește utilizarea instrucțiunii **PRINT USING** (v. linia 280) menționăm că limbajul ABASIC admite specificații de format atît pentru numere cît și pentru șiruri.

Simbolurile folosite pentru editarea valorilor numerice sînt:

"#", "–", "$$", "∧∧∧∧".

Aplicație. Scrieți un program BASIC care să tipărească următoarele combinații: 00, 01, 10, 11,

```
10 FOR I=0 TO 1
20    FOR J=0 TO 1
30       PRINT I; J
40    NEXT J
50 NEXT I
60 END
```

TEST

Precizați rezultatul execuției următorului program:

```
40 PRINT "50 FOR I=1 TO 5"
50 FOR I=1 TO 5
60    PRINT I;
70 NEXT I
80 PRINT
90 PRINT "100 FOR I=1 TO
   19 STEP 2"
100 FOR I=1 TO 19 STEP 2
110    PRINT I;
120 NEXT I
130 PRINT
140 PRINT "150 FOR I=345 TO
    282 STEP-9"
150 FOR I=345 TO 282 STEP-9
160    PRINT I;
170 NEXT I
180 PRINT
190 PRINT "200 FOR I=91.5
    TO 3 STEP - 15.7"
200 FOR I=91.5 TO 3 STEP - 15.7
210    PRINT I;
220 NEXT I
230 PRINT
240 PRINT "270 A=5 : B=45 : C=6"
250 PRINT "280 FOR I=A TO B
    STEP C"
270 A=5 : B=45 : C=6
280 FOR I=A TO B STEP C
290    PRINT I;
300 NEXT I
```

TEMA 4

□ **Răspundeţi prin DA sau NU la următoarele întrebări:**

– Blocul **PENTRU ... SFÎRŞIT** codifică o structură algoritmică de iteraţie cu număr cunoscut de paşi.
– Instrucţiunea **PENTRU** reprezintă punctul unic de intrare în blocul de iteraţie.
– Instrucţiunea **SFÎRŞIT** reprezintă punctul unic de ieşire din blocul de iteraţie.
– Instrucţiunile **FOR** şi **NEXT** apar în linii diferite în cadrul programului.
– Într-o buclă **FOR-NEXT** este obligatoriu ca variabila de control să ia valori începînd cu 1.
– Într-o instrucţiune **FOR** parametrul **STEP** este obligatoriu.
– În absenţa opţiunii **STEP** pasul de incrementare este 1.
– Variabila din instrucţiunea **NEXT** este diferită de variabila din instrucţiunea **FOR**.
– În BASIC-aMIC variabila din instrucţiunea **NEXT** poate să lipsească.
– În interiorul unei bucle **FOR-NEXT** din limbajul BASIC-aMIC se pot imbrica patru bucle.
– În BASIC-aMIC ciclurile suprapuse se pot intersecta.
– Dacă valoarea limită a variabilei de control este depăşită, execuţia buclei **FOR-NEXT** ia sfîrşit, următoarea instrucţiune ce urmează a fi executată fiind cea de după **NEXT**.
– Valoarea variabilei de control a unei bucle **FOR-NEXT** poate să descrească cu valori fracţionare.
– În limbajul BASIC-aMIC instrucţiunile **PRINT** şi **INPUT** nu pot fi combinate într-una singură – **INPUT**.

Secvenţele de instrucţiuni BASIC de mai jos sînt scrise corect:

```
– 10 FOR I=1 TO 20
       . . .
  40      I=I+1
       . . .
  90 NEXT I
```

```
– 10 FOR A=0 TO 40 STEP 4
       . . .
  80      FOR B=0 TO A
       . . .
 150      NEXT B
       . . .
 200 NEXT A
```

```
– 10 FOR I=1 TO M
       . . .
  40      FOR K=1 TO N
       . . .
  60      NEXT I
       . . .
  90 NEXT K
```

```
– 10 FOR A=0 TO 1 STEP 0.002
       . . .
  90      FOR A=0 TO 8 STEP 0.5
       . . .
 180 NEXT A
```

```
– 10 FOR B=I TO (C+D) STEP L
       . . .
  90 NEXT L
```

```
– 10 FOR A=B TO 4 * (C–D)
     STEP L
– 10 FOR I=0 TO – 80 STEP – 6
– 10 FOR I=J TO 80 STEP J
– 10 FOR A$=1 TO 100 STEP 2
– 30 FOR I=L TO A STEP I
– 10 FOR A=1 TO B STEP L
       . . .
  90 NEXT L
```

```
– 10 FOR I=1 TO A                100 FOR R=1 TO 50
   . . .                              . . .
  50   FOR J=1 TO 100            180 NEXT R
   . . .                              . . .
  80   FOR K=1 TO M              300 NEXT J
  90   NEXT K                         . . .
                                 900 NEXT I
```

– Instrucţiunea

```
10 INPUT "A, B, C"; A, B, C
```

este acceptată de limbajul BASIC-PRAE.

– În BASIC-PRAE variabila de control a buclei **FOR-NEXT** nu poate fi modificată.

– În BASIC-PRAE pot fi imbricate oricîte cicluri **FOR-NEXT** dorim cu condiţia ca acestea să nu se intersecteze.

Instrucţiunea

```
20 INPUT "A, B" ; A, B
```

nu este acceptată de calculatorul HC-85.

– În BASIC-SPECTRUM variabila de control a buclei **FOR-NEXT** trebuie să aibă numele format dintr-o singură literă.

– În BASIC-AMSTRAD orice neconcordanţă dintre numele variabilelor şi tipul datelor este semnalată prin mesajul

? Redo from start

– În BASIC-AMSTRAD, **NEXT**-ul poate fi . . . anonim (fără variabilă).

– În BASIC-AMSTRAD se pot realiza programe utilizînd clauzele de editare.

– Instrucţiunea care permite utilizatorului definirea unui format propriu de editare este **PRINT USING.**

– Indicatorul de cîmp "#" este folosit pentru a reprezenta poziţia fiecărei cifre a numărului.

– Instrucţiunea BASIC-AMSTRAD

```
10 PRINT USING "##.## " 4.3, 5.789
```

afişează: 4.30 5.79

– Instrucţiunea BASIC-AMSTRAD

```
10 PRINT USING "**$##.## " ; 8.25
```

afişează: ***$8.25

– instrucţiunea BASIC-AMSTRAD

```
10 PRINT USING " #### ↑↑↑↑ "; 777777
```

afişează .7779E06

– Instrucţiunea BASIC-COMMODORE

```
10 PRINT USING " ####", 3856
```

afişează 3856.

– În BASIC-80 caracterul "?" generat de instrucţiunea **INPUT** poate fi şters cu " , " sau " ; ".

– În BASIC-80 atribuirea variabilei de control lui **NEXT** este obligatorie.

– Instrucţiunea BASIC-80

```
10 PI=4 * ATN (1)
```

calculează numărul π.

– Indicatorul de cîmp " \ " este specific formatelor numerice de editare din BASIC-80.

– În BASIC-PLUS variabila de control a ciclului **FOR-NEXT** poate fi orice variabilă numerică simplă de tip întreg sau real.

– Variabila de control a unei bucle **FOR-NEXT** din BASIC-PLUS poate fi modificată în interiorul ciclului.

– Secvenţa de instrucţiuni BASIC

```
10 FOR I=1 TO 14
20     PRINT TAB (I); " * ";
30 NEXT I
```

tipăreşte un rînd alcătuit din 12 asteriscuri.

– Secvenţa de instrucţiuni BASIC-PLUS

```
20 PRINT " * " ; ; FOR I=1 TO 14
```

tipăreşte un rînd alcătuit din 13 asteriscuri.

– Secvenţa de instrucţiuni BASIC-PLUS

```
10 B$=" * "
20 PRINT B$ ; ; FOR I=1 TO 14
```

tipăreşte un rînd cu 14 asteriscuri.

– Secvenţa de instrucţiuni BASIC-PLUS

```
10 A=67801·78
20 PRINT USING ' $$##, ## · #', A
```

tipăreşte valoarea $67, 801.8.

– În limbajul ABASIC se pot imbrica mai multe cicluri **FOR-NEXT** cu terminarea pe un **NEXT** comun sau pe **NEXT**-uri diferite.

– Limbajul ABASIC admite specificaţii de format pentru valori numerice şi nenumerice.

☐ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Instrucţiunea **FOR** reprezintă punctul de .... în ciclu iar instrucţiunea NEXT reprezintă punctul de .... din ciclu.
b) Într-o instrucţiune FOR valoarea variabilei de control poate să crească fie cu valori .... fie cu valori ....
c) Rolul opţiunii **STEP** este ....
d) În BASIC-aMIC numărul maxim de cicluri imbricate este .... pe cînd în BASIC-PRAE numărul maxim de cicluri imbricate este ....
e) În limbajul BASIC-PLUS instrucţiunile **PRINT** şi **INPUT** .... fi cuplate într-una singură pe cînd în limbajul
pot/nu pot
BASIC-aMIC .... fi cuplate într-o instrucţiune **INPUT**.
pot/nu pot
f) În BASIC .... variabila de control a buclei **FOR-NEXT** poate fi modificată în interiorul instrucţiunii.
g) În BASIC .... ciclurile suprapuse se pot intersecta.
h) În BASIC-AMSTRAD variabila de control a buclei **FOR-NEXT** trebuie să aibă numele format din .... litere pe cînd în BASIC TIM S numele variabilei de control este format ....
i) Instrucţiunea **PRINT USING** poate fi utilizată în BASIC .... dar nu poate fi utilizată în BASIC ....
j) În BASIC .... atribuirea variabilei de control nu este obligatorie.
k) În BASIC-80 simbolurile folosite pentru editarea variabilelor numerice sînt ....
l) În BASIC-AMSTRAD simbolurile folosite pentru editarea valorilor numerice sînt .... iar în BASIC-PLUS sînt ....
m) Pentru editarea unui rînd cu 7 vaporaşe "<=>" se foloseşte secvenţa de instrucţiuni BASIC .... sau .... sau .... sau .... sau .... sau ....

☐ Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Aplicaţiile de la paginile 190, 195 (v. tabela de variabile şi schemele logice realizate).
b) Să se calculeze lungimile de undă ale spectrului emis de atomul de hidrogen, folosind formula lui Balmer.

c) Să se editeze în BASIC HC-85, TIMS, SPECTRUM textul de mai jos:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
I	N	V	A	T	A	M			B	A	S	I	C					.	.	.
I	N	V	A	T	A	M	B	A	S	I	C	.	.	.						
I	N	V	A	T	A	M		B	A	S	I	C		.	.	.				

d) Să se calculeze suma parțială a seriei

$$S=1-\frac{1}{4}+\frac{1}{9}+\ldots+(-1)^{N+1}/N^2, \text{ pentru N dat.}$$

e) Să se tipărească următorul triunghi al numerelor:

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
1   7   21  35  35  21  7   1
1   8   28  56  70  56  28  8   1
1   9   36  84  126 126 84  36  9   1
1   10  45  120 210 252 210 120 45  10  1
```

f) Să se calculeze rezistența echivalentă a N rezistențe legate în: a) serie; b) paralel.

g) Să se editeze următoarele blocuri de asteriscuri:

```
* * * *
* * * * * *

* * * * * *
* * * * * * * *
* * * * * * * * * *
* * * * * * * * * * * *

* * * * * * * *
* * * * * * * * * *
* * * * * * * * * * * *
* * * * * * * * * * * * * *
* * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * *
```

Schema logică este prezentată în figura 4.20.

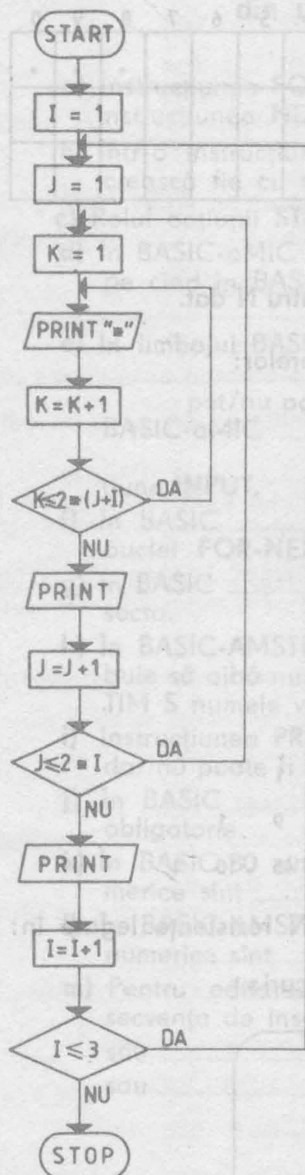


Fig. 4.20.

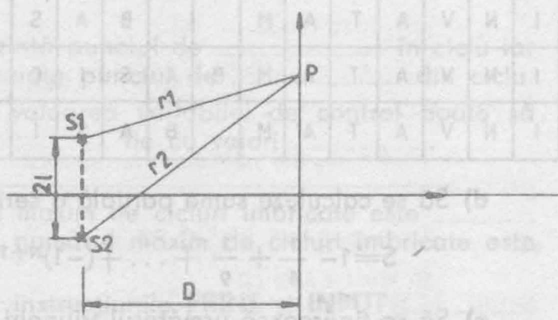


Fig. 4.21.

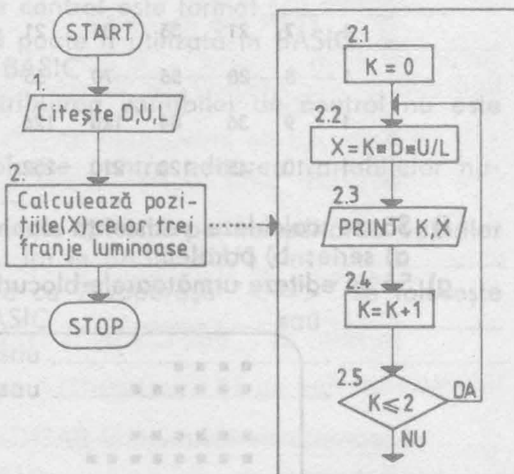


Fig. 4.22.

h) În experienţa lui Young se lucrează cu o radiaţie monocromatică cu $\lambda=6\cdot10^{-7}$ m. Distanţa dintre fante (v. fig. 4.21) este 1 mm, iar distanţa de la fante la ecran 3 m. Să se găsească poziţiile primelor trei franje.
Schema logică este prezentată în figura 4.22.

i) Să se calculeze abaterea standard a rezultatelor obţinute de un student la cursul de programarea calculatoarelor, ştiind că el a susţinut 8 teste. Se vor realiza tabela de variabile şi schema logică.

Indicaţie. Abaterea standard se calculează cu formula:

$$\varepsilon=\sqrt{\frac{\Sigma x^2-(\Sigma x)^2/n}{n-1}}, \text{ unde:}$$

n reprezintă numărul de note;
Σx este suma notelor;
Σx^2 este suma pătratelor notelor.

☐ **Se reia problema din tema precedentă complicînd-o** sub următorul aspect. Programul citeşte de la terminal cantităţile lunare de jucării pentru 500 de articole fabricate de întreprinderea „URSULEŢUL", calculează cantităţile totale trimestriale şi procentele corespunzătoare din cantitatea totală anuală.

☐ **Se reia problema din tema precedentă complicînd-o** după cum urmează. Programul citeşte de la terminal consumurile trimestriale pentru 1 000 de articole, calculează şi afişează procentele corespunzătoare consumurilor trimestriale din consumul total anual.

SOLUŢIA TEMEI 4

☐ Nu răspundem.

☐ Nu răspundem.

☐ **Programele BASIC sînt:**

b) Formula lui Balmer pentru determinarea lungimilor de undă a patru linii din vizibil ale atomului de hidrogen este:

$$\lambda=3645{,}6\,\frac{n^2}{n^2-4}\cdot 10^{-10}\,m$$

unde: λ este lungimea de undă;
n=3, 4, 5, 6.

Schema logică a programului este ilustrată în figura 4.23.

```
10 FOR N=3 TO 6
20    U=3645·6 * (N ↑ 2/(N ↑ 2–4))/10 ↑ 10
30    PRINT "N="; N; "U="; U
40 NEXT N
50 END
```

d)

```
10 S=0 : S1=1
20 INPUT "N="; N
30 FOR I=1 TO N
40      S=S+S1/I * I
50      S1=–S1
60 NEXT I
70 PRINT "S="; S
80 END
```

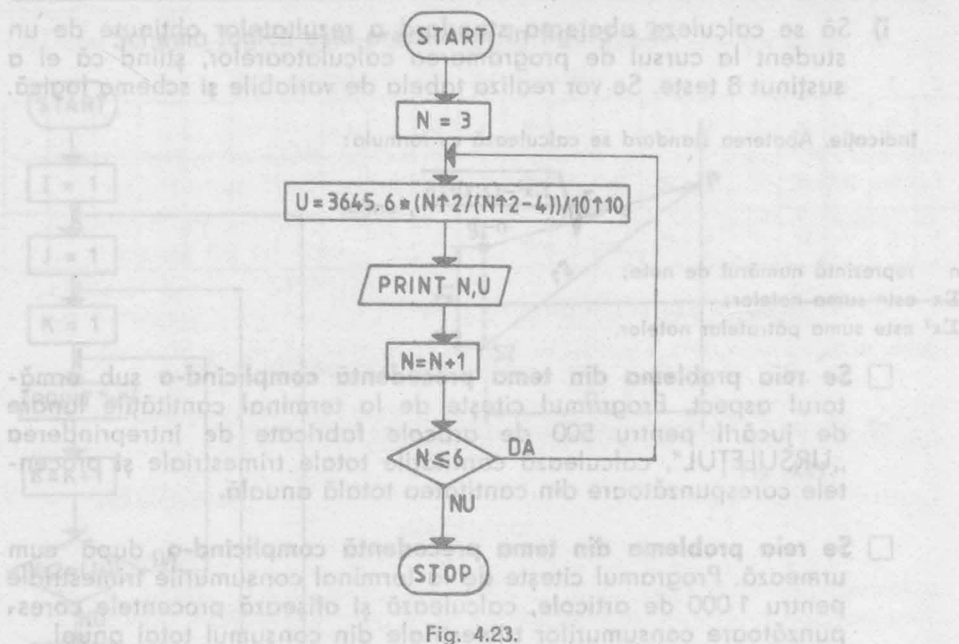


Fig. 4.23.

i)

```
10 B=0 : C=0
20 FOR I=1 TO 8
30    INPUT "NOTA="; N
40    B=B+N
50    C=C+N * N
60 NEXT I
70 E=((C-C/8)/7) ↑ .5
80 PRINT "E="; E
90 END
```

☐ **Se vor adăuga la programul din tema precedentă** (v. conversaţia 3, pag. 150) următoarele instrucţiuni:

```
1 FOR I=1 TO 500
```

(se copiază instrucţiunile din liniile 5–120).

```
125 NEXT I
130 END
```

☐ **Se vor adăuga la programul din tema precedentă** (v. conversaţia 3, pag. 153) următoarele instrucţiuni:

```
1 FOR I=1 TO 1000
```

(se copiază instrucţiunile din liniile 5–115).

```
115 NEXT I
120 END
```

Proiectarea şi realizarea unui program BASIC pentru calculul cantităţii de lichid, dintr-un număr variabil de rezervoare ale căror raze pot lua numai valori pozitive. Variabile indexate numerice. Vectori de date. Instrucţiunea DIM. Introducerea statică a datelor. Instrucţiunile READ şi DATA. Structura de iteraţie CÎT TIMP. Bucla WHILE-WEND. Structura de selecţie. Instrucţiunile IF-THEN şi IF-THEN-ELSE. Subprograme. Instrucţiunile GOSUB şi RETURN. JOCURI, APLICAŢII şi TESTE pentru cititor



□ De la problemă la program

Problema ce se pune seamănă cu precedentele, dar este mai complicată. Se citesc de la tastatură un număr oarecare (maximum 30) de valori ale razelor unor rezervoare cilindrice echilaterale. Pentru fiecare valoare a razei se calculează şi afişează cantitatea de benzină din rezervor. Programul verifică, totodată, ca valorile razelor rezervoarelor să fie numere pozitive. În caz de eroare se afişează un mesaj corespunzător. De asemenea, programul calculează şi afişează masa totală de benzină din rezervoare. În felul acesta, cu acelaşi program se pot rezolva probleme ce diferă prin numărul de rezervoare.

□ Analiza problemei

Problemele care se pun în această etapă privesc în special alcătuirea tabelei de variabile mai precis identificarea variabilelor de intrare ale programului.

Numărul rezervoarelor, necunoscut în problemă trebuie furnizat ca o variabilă de intrare (N) în momentul execuţiei programului, înaintea citirii datelor propriu-zise (razele rezervoarelor). Cît priveşte numărul variabil de raze, de asemenea necunoscut în momentul scrierii programului vom folosi ca variabilă de intrare un **vector** adică un tablou ce conţine un şir de variabile, avînd o dimensiune maximă cunoscută (30 în acest caz).

Variabila structurată de tip vector

Un vector poate fi imaginat ca o clădire cu mai multe apartamente. Dacă, de exemplu, numele clădirii este ZAMBILA şi ea dispune de 4 apartamente, atunci acele apartamente se vor identifica (într-o exprimare vectorială) prin ZAMBILA (1), ZAMBILA (2), ZAMBILA (3) ZAMBILA (4).

În informatică, se desemnează prin abuz de limbaj sub numele de vector, mulţimea {V(1). V(2), . . . , V(n)}, unde V(1), V(2), . . . , (Vn) sînt elementele vectorului. Aşadar, un vector este o mulţime de elemente identificate prin poziţia pe care o ocupă. Prelucrările ce se efectuează asupra elementelor unui vector sînt funcţie de valoarea unui indice de poziţie ($1 \leqslant i \leqslant n$). Acestuia i se fixează o valoare iniţială care, în general, corespunde primei poziţii, iar prelucrarea se efectuează cît timp valoarea acestui indice nu depăşeşte o valoare finală.

Pentru EXEMPLELE 5 vom defini variabila structurată de tip vector R (variabila de intrare, reprezentînd raza rezervorului) căreia îi asociem o variabilă de stare I semnificînd indicele de poziţie (indexul) al vectorului R cu valori între 1 şi N.

Specificaţiile de programare sînt prezentate în modulul de analiză structurată, figura 5.1.

FORMATUL DATELOR DE IEȘIRE

Nume program: EXEMPLELE 5 – PC, m, M, F

```
RAZA          MASA

XX            XXXX · XX
XX            XXXX · XX
.               .
.               .
.               .
- - - - - - - - - - - - - - - - - - - - - -
MASA TOTALĂ =  XXXXX · XX TONE
```

a)

TABELA DE VARIABILE

Nume program: EXEMPLELE 5 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
R : vectorul razelor rezervorului	I : indicele de poziție (indexul)	

b)

TABELA DE VARIABILE

Nume program: EXEMPLELE 5–PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
N: număr de rezervoare	V: volumul rezervorului S: masa totală de benzină M: masa de benzină din rezervor	M: masa de benzină din rezervor S: masa totală de benzină

c)

Fig. 5.1. a-d. Modulul de analiză structurată: *a)* formatul datelor de ieșire; *b)* tabela de variabile; *c)* tabela de variabile (completare);

SPECIFICAŢII DE PROGRAMARE

☐ Nume program: EXEMPLELE 5 – PC, m, M, F

Descrierea programului

Programul calculează şi afişează cantitatea de benzină (S) dintr-un număr oarecare (N) de rezervoare cilindrice echilaterale (acest număr este furnizat ca parametru). Programul citeşte valorile razelor rezervoarelor într-un vector (R) printr-o procedură de introducere dinamică a datelor (razele cu valori negative nu se iau în considerare). Programul citeşte densitatea benzinei (D) printr-o procedură de introducere statică a datelor.

Intrări
Se introduc de la tastatură numărul rezervoarelor cilindrice echilaterale şi valorile razelor acestora.

Ieşiri
Masa de benzină (M) din fiecare rezervor şi masa totală (S) de benzină.

Lista de funcţiuni ale programului
1. Citeşte număr rezervoare (N)
2. Iniţializează variabila S
3. Afişează un rînd cu simbolul "="
4. Afişează "raza, masa"
5. Pentru fiecare rezervor:
 5.1. Citeşte valoarea razei unui rezervor
 5.2. Validează datele introduse după cum urmează: pentru fiecare articol se verifică dacă valoarea razei citite este negativă. Programul afişează mesajul "Raza negativă".
6. Pentru fiecare rezervor:
 6.1. Calculează volumul rezervorului
 6.2. Calculează masa de benzină din rezervor
 6.3. Însumează M în S
 6.4. Afişează R, M
7. Afişează masa totală de benzină
8. Stop

d)

d) specificaţii de programare.

☐ Proiectarea programului

Diagrama de structură

Pentru trasarea diagramei de structură s-a procedat la identificarea, mai întîi, a principalelor funcţiuni de prelucrare apoi la alocarea acestor funcţiuni modulelor de program (v. fig. 5.2 (a)), după aceea trecîndu-se la reprezentarea diagramei de structură (v. fig. 5.2 (b)).

Vom decupa din această diagramă blocul de iteraţie PREL-VALID ale cărui funcţiuni sînt:

- citeşte raza;
- afişează raza;
- validează datele introduse şi afişează mesaj în caz de eroare (raza negativă).

Observaţie. Lista completă de funcţiuni este prezentată în modulul de analiză structurată, fig. 5.1 (d).

De notat că blocul PREL-VALID (nivelul 1) conţine şi el un bloc de iteraţie-VALID (nivelul 2) care se execută **cît timp** raza rezervorului este negativă (v. lista de funcţiuni ale modulului).

Spre deosebire de blocul PREL-VALID, blocul VALID reprezintă o **structură de iteraţie condiţionată anterior ce se execută cît timp condiţia ("Raza este negativă") este adevărată.**

Remarci

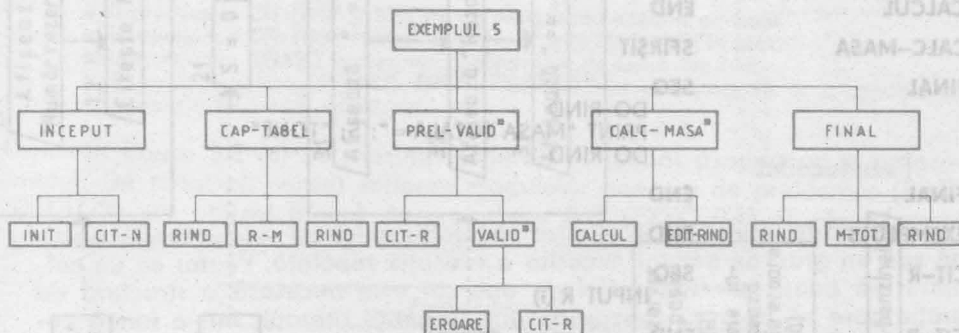
- Diagrama de structură utilizează două module comune: RIND (v. blocurile CAP-TABEL, FINAL) şi CIT-R (v. blocurile PREL-VALID, VALID).
- Lista de funcţiuni a modulului CALCUL implică prelucrări de vectori.
- Blocul PREL-VALID reprezintă o structură de iteraţie cu număr cunoscut de paşi (N).
- Blocul VALID reprezintă o .structură de iteraţie condiţionată anterior.

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 5 – PC, m, M, F

Modul	Funcţiuni
INIT	2
CIT–N	1
RIND	3
R–M	4
PREL–VALID	5
VALID	5.2, 5.1
EROARE	5.2
CALCUL	6.1, 6.2
M–TOT	6.3
FINAL	3, 7, 8
CIT-R	5.1
EDIT–RIND	6.4

a)



b)

Fig. 5.2. a-d. Modulul de proiectare structurată: *a)* alocarea funcţiunilor de prelucrare; *b)* diagrama de structură;

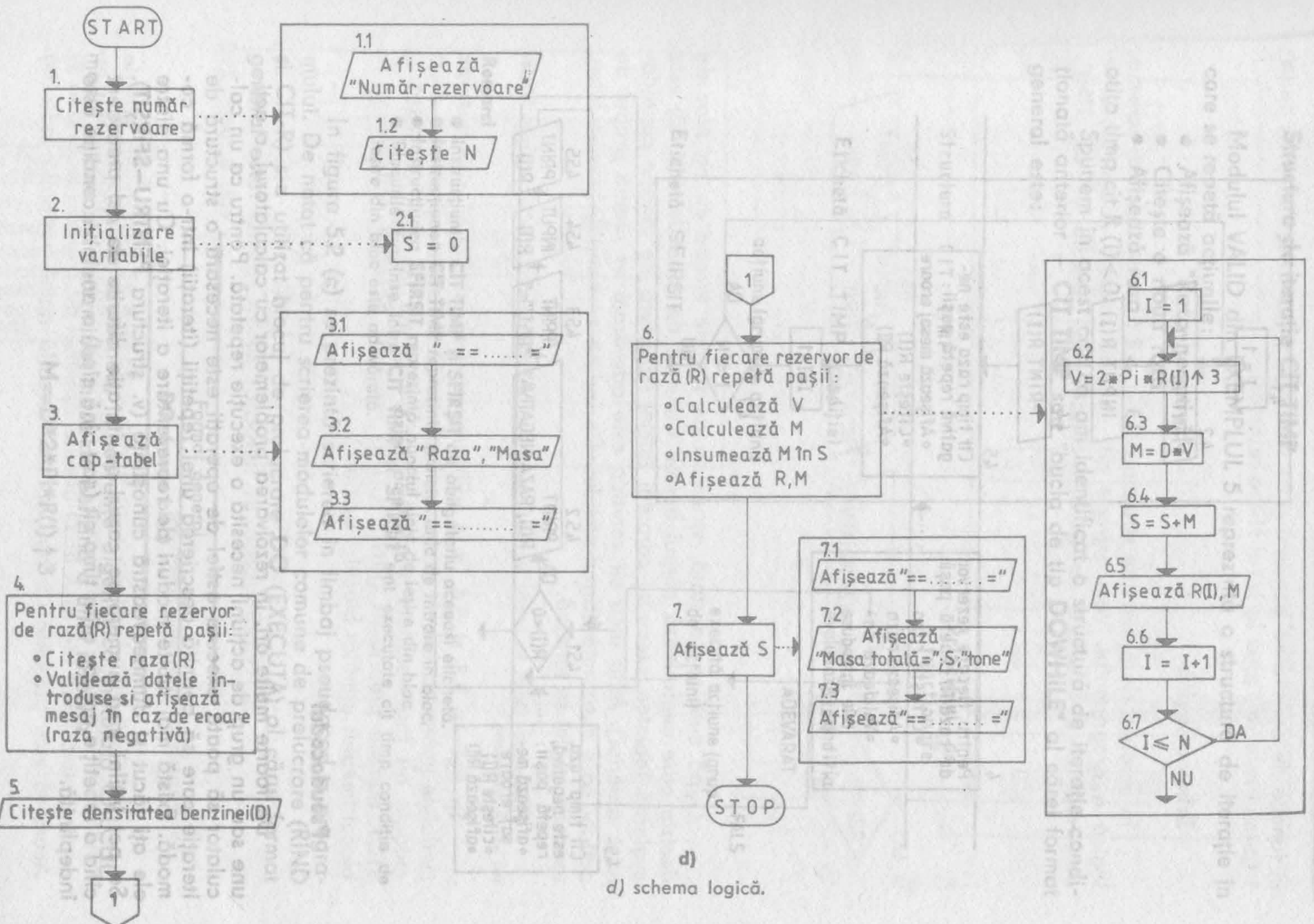
PSEUDOCODUL

Nume program: EXEMPLELE 5 – PC, m, M, F

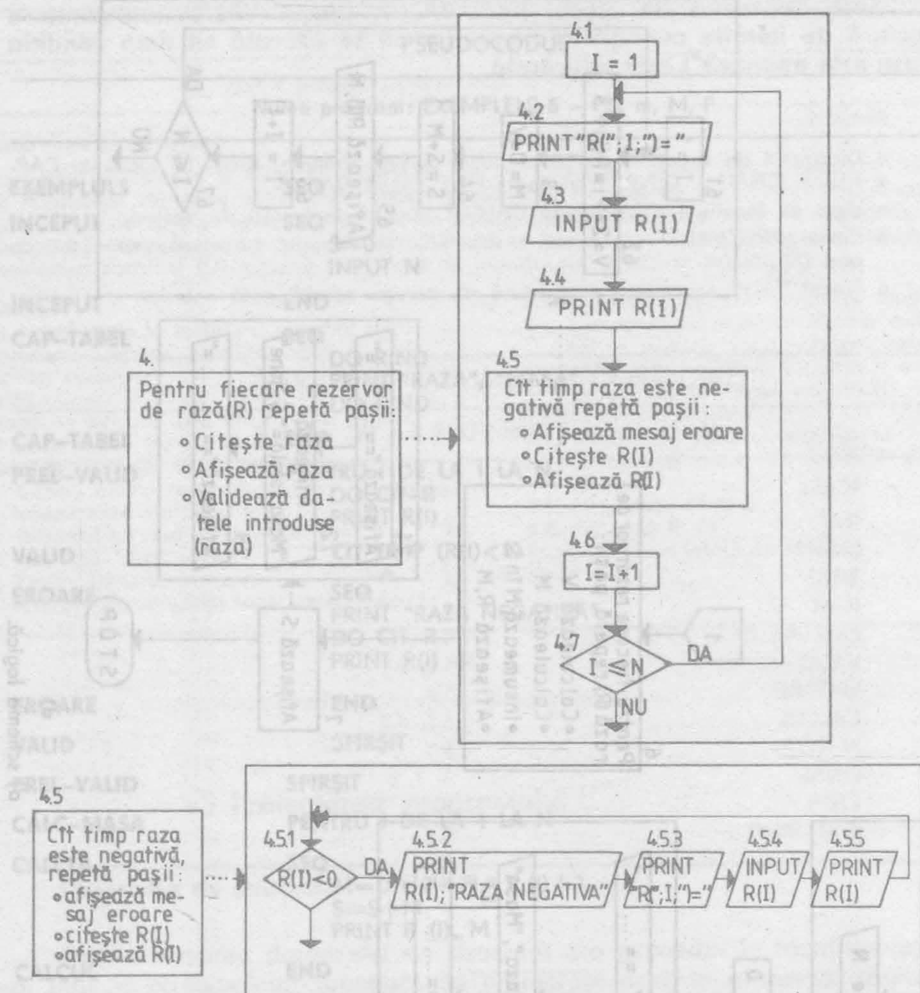
```
EXEMPLUL5      SEQ
INCEPUT        SEQ
                 S=0
                 INPUT N
INCEPUT        END
CAP-TABEL      SEQ
                 DO RIND
                 PRINT "RAZA", "MASA"
                 DO RIND
CAP-TABEL      END
PREL-VALID     PENTRU I DE LA 1 LA N
                 DO CIT-R
                 PRINT R(I)
VALID          CIT TIMP (R(I)<0)
EROARE           SEQ
                 PRINT "RAZA NEGATIVĂ"
                 DO CIT-R
                 PRINT R(I)
EROARE           END
VALID            SFIRŞIT
PREL-VALID     SFIRŞIT
CALC-MASA      PENTRU I DE LA 1 LA N
CALCUL         SEQ
                 M=D * 2 * PI * R (I) ↑ 3
                 S=S+M
                 PRINT R (I), M
CALCUL         END
CALC-MASA      SFIRŞIT
FINAL          SEQ
                 DO RIND
                 PRINT "MASA TOTALĂ="; S; "TONE"
                 DO RIND
FINAL          END
EXEMPLUL5      END
CIT-R          SEQ
                 INPUT R (I)
CIT-R          END
```

c)

c) pseudocodul;



d) schema logică.



d) schema logică.

Pseudocodul

De foarte multe ori, în rezolvarea problemelor cu calculatorul, o acţiune sau un grup de acţiuni necesită o execuţie repetată. Pentru ca un calculator să poată executa astfel de operaţii este necesară o structură de iteraţie care să permită descrierea unei repetiţii (iteraţii) într-o formă comodă. Există mai multe moduri de reprezentare a iteraţiei. Cu una dintre ele aţi făcut şi dumneavoastră cunoştinţă (v. structura **PENTRU–SFîRŞIT**). Se pot întîlni însă numeroase cazuri de iteraţie diferite de cel precedent cînd o repetiţie are loc atîta timp cît (sau pînă cînd) o anumită condiţie este îndeplinită.

Structura de iteraţie CÎT TIMP

Modulul VALID din EXEMPLUL 5 reprezintă o structură de iteraţie în care se repetă acţiunile:

- Afişează "Raza negativă"
- Citeşte o nouă rază
- Afişează raza

atîta timp cît R (I)<0.

Spunem în acest caz că am identificat o structură de iteraţie condiţionată anterior – **CÎT TIMP** sau "bucla de tip **DOWHILE**" al cărei format general este:

Structura CIT TIMP	Principiul de funcţionare
Etichetă CIT TIMP (condiţie) acţiune (grup de acţiuni) Etichetă SFIRSIT	evaluează condiţia ADEVARAT execută acţiune (grup de acţiuni) FALS

Remarci

- Instrucţiunile **CÎT TIMP** şi **SFÎRŞIT** au obligatoriu aceeaşi etichetă.
- Instrucţiunea **CÎT TIMP** reprezintă punctul unic de intrare în bloc.
- Instrucţiunea **SFÎRŞIT** reprezintă punctul unic de ieşire din bloc.
- Blocurile cuprinse între **CÎT TIMP** şi **SFÎRŞIT** sînt executate cît timp condiţia de ieşire din bloc este adevărată.

În figura 5.2 (c) se prezintă scrierea în limbaj pseudocod a programului. De notat că pentru scrierea modulelor comune de prelucrare (RÎND şi CIT-R) s-a utilizat blocul de prelucrare **DO** (EXECUTA) al cărui format general este:

Format general
DO ⟨nume bloc⟩

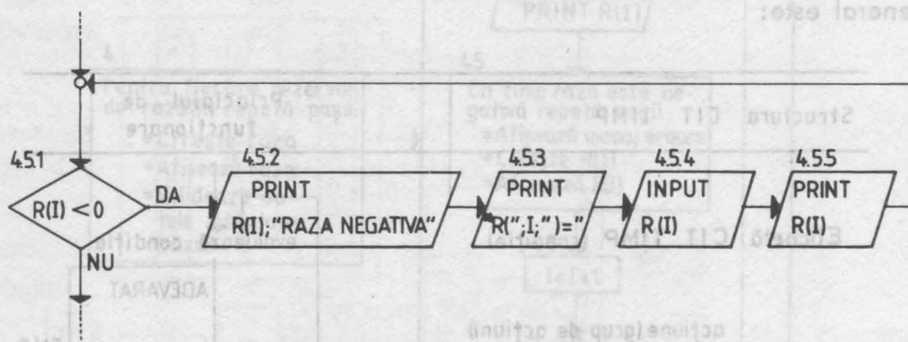
Remarcaţi, de asemenea, în secvenţa algoritmică intitulată CALCUL, modul în care s-a determinat masa de benzină a unui rezervor:

M=D*2*PI*R(I) ↑ 3

în funcţie de densitatea benzinei şi de raza R(I), pentru I avînd valori de la 1 (primul rezervor) pînă la N (al N-lea rezervor).

Schema logică

În modulul de proiectare structurată, fig. 5.2 (d) este ilustrată schema logică a programului. Vom extrage din această schemă numai blocurile care reprezintă structura de iteraţie condiţionată anterior – **CÎT TIMP.**



După cum se constată, structura de iteraţie începe cu un bloc de decizie care controlează corpul buclei. Condiţia de repetiţie (R(I)<0) este evaluată înaintea execuţiei corpului buclei (structura de iteraţie condiţionată anterior). Atita timp cît condiţia este adevărată se repetă grupul de acţiuni care constituie corpul ciclului. De îndată ce condiţia devine falsă, execuţia buclei se termină.

Observaţie. Dacă la prima evaluare condiţia nu este adevărată, atunci corpul buclei nu va fi executat.

În figurile 5.3 şi 5.4 "prin utilizarea simbolului "‖ ‖" sînt ilustrate modulele independente de program RÎND şi CIT-R. Remarcaţi că cele două module au fost detaliate separat, avînd menţionat în locul lui START şi STOP numele propriu-zis (RÎND, CIT-R) respectiv **RETURN** (întoarcerea la blocul imediat următor blocului apelant).

□ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 210

Variabile indexate numerice

Noţiunea următoare pe care o vom introduce vă cere multă atenţie. Pînă acum am folosit numai variabile simple. O variabilă simplă este formată dintr-o literă (de la A la Z) urmată opţional de o cifră (de la 0 la 9). Acum vă vom prezenta un nou tip de variabilă numit variabila indexată.

4\.
Pentru fiecare rezervor de rază(R) repetă paşii:
• Citeşte raza (R)
• Afişează raza (R)
• Validează datele introduse şi afişează mesaj de eroare (rază negativă)

4.1 I = 1
4.2 Execută CIT-R
4.3 R(I)<0
DA
4.4 PRINT "R(";I;")"; R(I);" Raza negativă"
4.5 Execută CIT-R
NU
4.6 I=I+1
4.7 I≤N
DA
NU

CIT-R
4.2.1 Afişează "R(";I;")="
4.2.2 INPUT R(I)
4.2.3 PRINT R(I)
RETURN

Fig. 5.3.

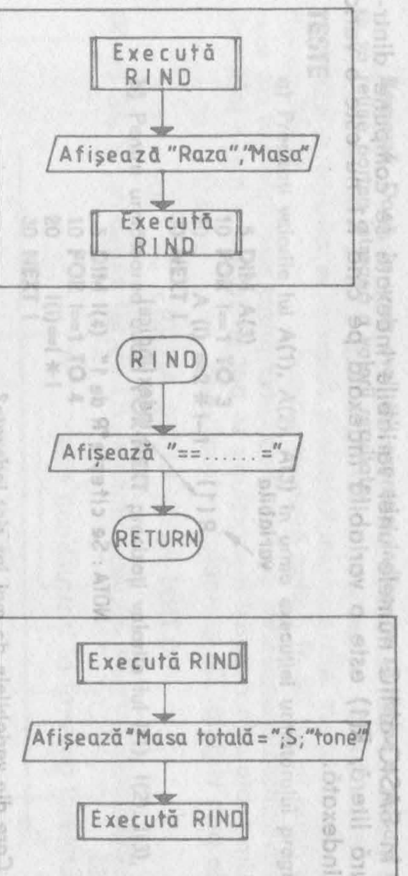


Fig. 5.4.

În BASIC-aMIC numele unei variabile indexate se compune dintr-o singură literă, R(1) este o variabilă indexată pe cînd R1 nu este o variabilă indexată.

variabila

R(1)

index (indice)

NOTA: Se citeşte "R de 1"

TEST

Care din variabilele de mai jos sînt indexate?

A (2); A; A 2; B (30); M.

R. A (2) şi B (30)

De notat că A, A2 şi A(2) reprezintă trei variabile diferite. Dumneavoastră le puteţi confunda, dar calculatorul le va recunoaşte ca variabile distincte.

R(1)	
R(2)	8
R(3)	
R(4)	
R(5)	3
R(6)	
R(7)	11
R(8)	
R(9)	

Observaţie. Într-un program BASIC-aMIC pot exista variabile simple numerice şi variabile indexate cu acelaşi nume.

Ca şi variabilele simple folosite pînă acum, variabila indexată ocupă, de asemenea, o locaţie în memoria calculatorului. Un set de variabile indexate cu o singură dimensiune reprezintă un vector.

Să presupunem acum că dumneavoastră sînteţi ... calculatorul şi R(5)=3.

Cu alte cuvinte, luaţi, vă rugăm creionul şi puneţi numărul 5 în cutia alăturată lui R(5). Realizaţi acelaşi lucru şi cu R(2)=8. Care va fi valoarea lui R(7) dacă

R(7)=R(2)+R(5)

Verificaţi-vă răspunsul cu rezultatul din figură.

Ce este aşa de minunat şi de misterios în legătură cu aceste variabile? Iată explicaţia: variabilelor indexate li se pot ataşa anumite valori. De exemplu, variabila R(I) care are drept index tot o variabilă.

Dacă I=1 atunci R(I) este R(1);
Dacă I=2 atunci R(I) este R(2);
Dacă I=4 atunci R(I) este R(4)

Indicele poate fi, de asemenea, o expresie, de exemplu R(I+8). Dacă I=5 atunci R(I+8) este R(13).

Remarcă. Dacă, în urma evaluării expresiei indice, nu se obţine o valoare întreagă, se va reţine partea întreagă a valorii rezultate.

TESTE

a) Precizaţi valorile lui A(1), A(2), A(3) în urma execuţiei următorului program:

```
5 DIM A(3)
10 FOR I=1 TO 3
20    A (I) =2 * I–1
30 NEXT I
```

b) Pentru următoarea buclă **FOR-NEXT** precizaţi valorile lui I(1), I(2), I(3), I(4).

```
5 DIM I(4)
10 FOR I=1 TO 4
20    I(I)=I * I
30 NEXT I
```

Introducerea dinamică a vectorilor de date

Cum pot fi folosite cel mai uşor variabilele indexate în limbajul BASIC? Un prim mod de a le utiliza este cel al creării unui vector de date cu ajutorul instrucţiunii **INPUT** plasate într-o buclă **FOR-NEXT**. Pentru a ilustra acest caz, vă invităm să urmăriţi instrucţiunile 70, 80, 90, 100 şi 170 (numai pe acestea!) din program:

```
70 FOR I=1 TO N
80    PRINT "R (" ; I ; ") = " ;
90    INPUT R (I)
100   PRINT R (I)
.     .
.     .
.     .
170 NEXT I
```

De notat că în timpul execuţiei acestei secvenţe de program se vor citi N valori (corespunzătoare razelor celor N rezervoare cilindrice echilaterale) ce se vor atribui variabilelor R(1), R(2), . . . , R(N).

Totul pare normal, dar cu toate acestea ceva este în neregulă. Nu putem executa secvenţa de instrucţiuni BASIC (70–140) fără a-i preciza calculatorului valoarea lui N (numărul de rezervoare) dar şi spaţiul de memorie pe care trebuie să-l rezerve pentru datele ce aparţin vectorului R(I). Vom realiza acest lucru adăugînd programului secvenţa de instrucţiuni:

```
20 PRINT "NUMĂR REZERVOARE";
30 INPUT N
40 PRINT N
60 DIM R (30)
```

DIM: instrucţiune de dimensionare a tablourilor

DIM face o rezervare de memorie pentru date, dar nu determină nici un fel de operaţii (aritmetice sau de alt fel). Instrucţiunea **DIM** este foarte des utilizată în limbajul BASIC şi totuşi foarte mulţi o evită întrucît nu înţeleg exact care este rolul său.

În cazul nostru în loc să se folosească N variabile pentru valorile razelor rezervoarelor cilindrice echilaterale s-a folosit un vector cu N elemente recunoscut sub numele R: prima valoare se află în primul element, a doua în cel de-al doilea element ş.a.m.d.

Într-o instrucţiune **DIM** se specifică între paranteze numărul de elemente din tablou. În aceeaşi instrucţiune se pot declara mai multe tablouri, separate prin virgulă. De exemplu, instrucţiunea

10 **DIM** A(30), A1(20), A2(50)

descrie trei vectori: A cu 30 de elemente, A1 cu 20 elemente, şi A2 cu 50 elemente.

Formatul general al instrucţiunii este:

Format general
⟨nr. linie⟩ **DIM** ⟨variabilă⟩ (⟨dim1⟩ [,⟨dim2⟩]) {, ⟨variabilă⟩ (⟨dim1⟩ [,⟨dim2⟩])}

în care:

⟨variabilă⟩ indică numele tabloului. El poate fi orice nume de variabilă indexată admisă de limbaj;

⟨dim1⟩, ⟨dim2⟩ reprezintă dimensiunile tabloului. Ele pot fi constante, variabile numerice simple (de tip întreg) sau expresii aritmetice.

Remarci

- Dimensiunea unui tablou are valori cuprinse în intervalul [1, 254].
- Tablourile numerice pot avea una sau două dimensiuni.
- Indicele unei variabile indexate trebuie să aibă valori cuprinse în intervalul [1,⟨dimensiune declarată în **DIM**⟩].
- La execuţia instrucţiunii **DIM** pentru tablourile numerice, se alocă cîte 4 octeţi pentru fiecare element al unei variabile tablou.
- În cazul utilizării unor tablouri mai există situaţii cînd la execuţia programului se afişează mesajul de eroare MEMORY FULL indicînd faptul că nu mai există memorie suficientă pentru alocarea tablourilor.

Introducerea statică a vectorilor de date

Pînă acum am introdus datele (în memoria calculatorului) în mod dinamic (v. instrucţiunea **INPUT**). Mai există o posibilitate pe care ne-o oferă limbajul BASIC – introducerea datelor în mod static. Pentru aceasta se înlocuieşte **INPUT** cu instrucţiunile **READ** şi **DATA**. **READ** citeşte datele conţinute de o linie **DATA**.

Formatul instrucţiunilor este:

Format general
⟨nr. linie⟩ **READ** ⟨lista variabile⟩

Format general
⟨nr. linie⟩ **DATA** ⟨listă constante⟩

Ce se "pune" în instrucţiunea **DATA**? Tot ceea ce doriţi: cifre, litere sau litere şi cifre. Pentru citirea şi utilizarea acestor date (listă constante) de către calculator, ele vor fi afectate unei liste de variabile din instrucţiunea **READ**.

În figura 5.5 se prezintă mecanismul de introducere statică a datelor.

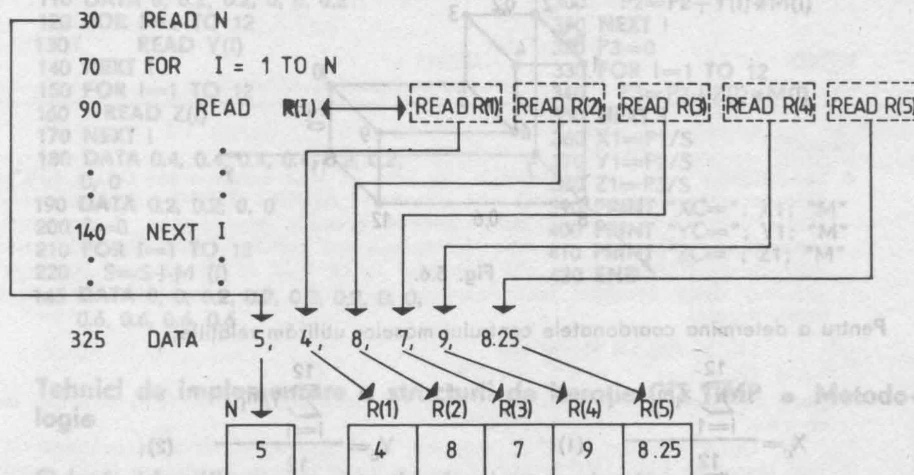


Fig. 5.5. Modul de utilizare a instrucţiunilor **READ** şi **DATA**.

Remarci

- Instrucţiunea **READ** are acelaşi efect cu **INPUT**, cu deosebirea că datele nu sînt introduse de la tastatură, ci sînt citite dintr-un bloc de date, definit cu instrucţiunea **DATA**.
- O instrucţiune **DATA** nu este asociată unei instrucţiuni **READ**, ci toate instrucţiunile **DATA** sînt tratate ca şi cum ar forma un bloc de date.
- Instrucţiunea **DATA** poate apare oriunde în cadrul programului.
- Pentru ţinerea evidenţei constantelor citite în instrucţiunile **DATA**, interpretorul BASIC-aMIC foloseşte un indicator.

În cadrul programului BASIC-aMIC instrucţiunile **READ-DATA** au fost întrebuinţate pentru citirea valorii densităţii benzinei în mod static.

```
190 READ D
.
.
.
350 DATA .7
```

În urma execuţiei instrucţiunii din linia 190 lui D i se atribuie valoarea 0.7.

TEST

Precizaţi rezulatul execuţiei următorului program:

```
 3 DIM A(3)
 5 READ N
10 FOR I=1 TO N
20    READ A(I)
30       PRINT A(I)
40 NEXT I
50 DATA 3, 7, 8, 9
60 END
```

Aplicaţie. Se consideră sistemul de puncte materiale din figura 5.6, avînd masele $m_1=m_5=m_{12}=0{,}004$ kg; $m_3=m_8=m_{10}=0{,}006$ kg; $m_2=m_6=m_{11}=0{,}008$ kg; $m_4=m_7=m_9=0{,}002$ kg. Distanţele dintre punctele materiale sînt exprimate în metri. Să se determine coordonatele centrului maselor.

Fig. 5.6.

Pentru a determina coordonatele centrului maselor utilizăm relaţiile:

$$X_c=\frac{\sum_{i=1}^{12} x_i m_i}{\sum_{i=1}^{12} m_i} \quad (1); \qquad Y_c=\frac{\sum_{i=1}^{12} y_i m_i}{\sum_{i=1}^{12} m_i} \quad (2);$$

$$Z_c=\frac{\sum_{i=1}^{12} z_i m_i}{\sum_{i=1}^{12} m_i} \quad (3).$$

şi completăm tabelul de mai jos:

Nr. crt.	m_i	x_i	y_i	z_i
1	0,004	0,2	0	0,4
2	0,008	0	0	0,4
3	0,006	0	0,2	0,4
4	0,002	0,2	0,2	0,4
5	0,004	0	0,2	0,2
6	0,008	0,2	0,2	0,2
7	0,002	0	0	0
8	0,006	0,2	0	0
9	0,002	0,2	0,6	0,2
10	0,006	0	0,6	0,2
11	0,008	0	0,6	0
12	0,004	0,2	0,6	0

```
10 REM CENTRUL MASELOR
20 DIM M(12), X(12), Y(12), Z(12)
30 FOR I=1 TO 12
40 READ M(I)
50 NEXT I
```

```
60 DATA 0.004, 0.008, 0.006, 0.002
   0.004, 0.008, 0.002, 0.006,
   0.002, 0.006, 0.008, 0.004
70 FOR I=I TO 12
80   READ X(I)
90 NEXT I
100 DATA 0.2, 0, 0. 0.2, 0, 0.2
110 DATA 0, 0.2, 0.2, 0, 0, 0.2
120 FOR I=1 TO 12
130      READ Y(I)
140 NEXT I
150 FOR I=1 TO 12
160   READ Z(I)
170 NEXT I
180 DATA 0.4, 0.4, 0.4, 0.4, 0.2, 0.2,
    0, 0
190 DATA 0.2, 0.2, 0, 0
200 S=0
210 FOR I=1 TO 12
220   S=S+M (I)
145 DATA 0, 0, 0.2, 0.2, 0.2, 0.2, 0, 0,
    0.6, 0.6, 0.6, 0.6
230 NEXT I
240 P1=0
250 FOR I=1 TO 12
260      P1=P1+X (I) * M (I)
270 NEXT I
280 P2=0
290 FOR I=1 TO 12
300    P2=P2+Y(I)*M(I)
310 NEXT I
320 P3=0
330 FOR I=1 TO 12
340    P3=P3+Z(I)*M(I)
350 NEXT I
360 X1=P1/S
370 Y1=P2/S
380 Z1=P3/S
390 PRINT "XC="; X1; "M"
400 PRINT "YC="; Y1; "M"
410 PRINT "ZC="; Z1; "M"
420 END
```

Tehnici de implementare a structurii de iteraţie CÎT TIMP • Metodologie

Odată identificate structurile fundamentale de program (secvenţa, selecţia, iteraţia) se pune problema implementării lor, în cazul nostru în limbajul BASIC. Iată posibilităţile principale:

a) utilizarea unor interpretoare (compilatoare) speciale de BASIC structurat; **b)** utilizarea precompilatoarelor, în care pornindu-se de la un limbaj auxiliar de descriere a structurilor (pseudocod) se generează instrucţiuni BASIC standard; **c)** simularea structurii în BASIC standard cu respectarea unei anumite metodologii.

În acest capitol vom trata implementarea structurii fundamentale de program **CÎT TIMP**, pe calculatorul aMIC, utilizînd limbajul "BASIC structurat" care poate fi prelucrat de un interpretor corespunzător: BASIC-AMSTRAD, BASIC-80, ABASIC, GWBASIC.

Pe calculatoarele personale aMIC, PRAE, HC-85, COMMODORE cît şi pe minicalculatoarele INDEPENDENT, CORAL nu există un astfel de interpretor, implementarea structurii descrise putîndu-se efectua printr-un preprocesor sau prin "traducere cu mîna".

Codificarea în BASIC structurat a buclei de tip **CÎT TIMP** este:

```
WHILE ⟨condiţie⟩
   ⟨corpul buclei⟩
WEND
```

Remarci

- **WHILE** indică punctul de intrare în ciclu.
- **WEND** indică punctul de ieşire din ciclu.
- ⟨condiţie⟩ defineşte condiţia care controlează corpul buclei.
- pot exista oricîte nivele de cicluri **WHILE-WEND**.
- fiecare **WEND** închide cel mai apropiat **WHILE**.

Simularea iterației

BASIC Structurat → translatare → BASIC standard	
WHILE ⟨condiție⟩	e1 IF ⟨complementul condiției⟩ THEN e2
⟨corpul buclei⟩	⟨corpul buclei⟩
WEND	GOTO e1 e2 ⟨instrucțiunea următoare⟩

Remarci

- Semnificația instrucțiunii **IF–THEN** din linia e1 este următoarea: **DACA (IF)** condiția (complementul condiției) este adevărată **ATUNCI (THEN)** calculatorul merge ("sare") la linia e2 după care continuă execuția programului. Prima parte a instrucțiunii **(IF)** pune condiția (complementul condiției) iar cealaltă parte a instrucțiunii **(THEN)** spune calculatorului la ce linie e2 să se ducă (ce linie să urmeze) dacă condiția (complementul condiției) este adevărată. De exemplu, instrucțiunea

```
110 IF R(I)>0 THEN 170
120
```

spune calculatorului să meargă la linia 170 *dacă* valoarea lui R (I) este mai mare ca zero.

Dacă valoarea lui R (I) este mai mică sau egală cu zero, calculatorul nu mai merge la linia 170. El pur și simplu trece la linia următoare (120), continuînd în acest fel execuția programului.

- Instrucțiunea **GOTO** spune calculatorului să facă un "salt" înainte sau înapoi, în cadrul programului la linia a cărei etichetă e1 este indicată după **GOTO**.

De exemplu, instrucțiunea **GOTO** din linia 160

```
110 IF R(I)>0 THEN 170
160     GOTO    110
```

spune calculatorului să se ducă de la linia 160 la linia 110 (salt necondiționat) și apoi să continue cu instrucțiunea din linia 110, 120 etc.

Metodologie pentru simularea iterației CÎT TIMP

Putem formula o metodologie de translatare (din BASIC structurat în BASIC-aMIC Standard) respectînd următorii pași:

1. Alegem două numere de linie e1 și e2 nefolosite în program pentru instrucțiunile **GOTO** și **IF**.
2. Folosind complementul condiției $\overline{C}$ scriem instrucțiunea BASIC e1 **IF** $\overline{C}$ **THEN** e2
3. Listăm instrucțiunile ce urmează a fi incluse în corpul buclei
4. Scriem instrucțiunile

GOTO e1

e2 ⟨instrucțiunea următoare⟩,
urmînd imediat corpul buclei.

Înapoi la program

Urmăriți pentru început, în tabelul 5.1 simularea iterației **CIT TIMP** în limbajul BASIC-aMIC. Remarcați corespondența dintre **WHILE–IF THEN** și **WEND–GOTO.**

Tabelul 5.1

BASIC structurat	BASIC-aMIC (Standard)
110 **WHILE** R(I)<0	110 **IF** R(I)>0 **THEN** 170
120 **PRINT** "R("; I; ")="; R(I); "RAZA NEGATIVĂ"	120 **PRINT** "R("; I; ")="; R(I); "RAZA NEGATIVĂ"
130 **PRINT** "R("; I; ")=";	130 **PRINT** "R("; I; ")=";
140 **INPUT** R(I)	140 **INPUT** R(I)
150 **PRINT** R(I)	150 **PRINT** R(I)
160 **WEND**	160 **GO TO** 110
170	170

În figura 5.7 puteţi urmări, de asemenea, întregul mecanism de funcţionare al instrucţiunilor **FOR, IF–THEN** şi **GOTO** aplicat programului BASIC-aMIC.

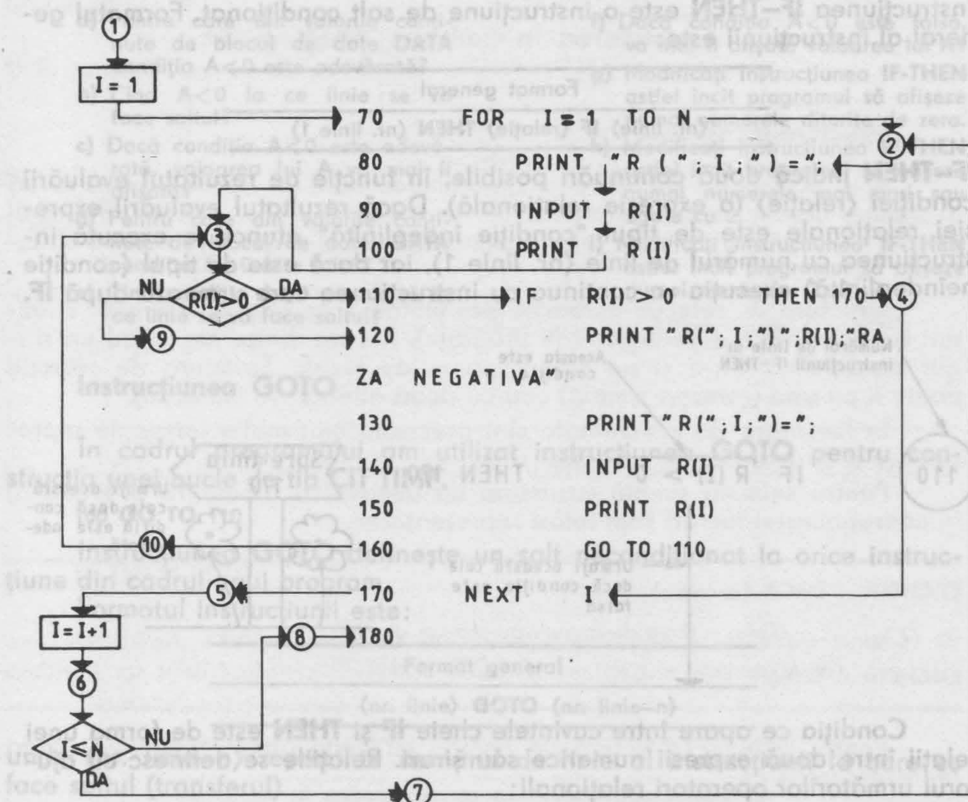


Fig. 5.7. Mecanismul de funcţionare a instrucţiunlor **FOR, IF-THEN** şi **GOTO**.

1) Se iniţializează variabila de control (I) a buclei **FOR**; 2) Se execută instrucţiunile din liniile 80, 90, 100; 3) Se testează condiţia R(I)>0; 4) Dacă condiţia este adevărată se execută saltul la linia 170; 5) Se incrementează valoarea lui I; 6) Se testează condiţia I≤N; 7) Dacă condiţia este adevărată se reia ciclul de la pasul 2; 8) Dacă condiţia (I≤N) nu este adevărată, se execută instrucţiunea din linia 180; 9) Dacă condiţia R(I) nu este adevărată, se execută instrucţiunea imediat următoare după **IF** (linia 120) urmată de instrucţiunile din liniile 130, 140 şi 150; 10) Se execută saltul necondiţionat la instrucţiunea din linia 110.

Structuri alternative. Instrucţiunea IF-THEN

În mod normal, paşii unui algoritm sînt realizaţi în ordinea specificării lor. Totuşi succesiunea acestora este determinată de structura datelor de intrare. În funcţie de valorile acestora se iau anumite decizii. Rezultatul evaluării determină ce paşi ai algoritmului urmează să se execute. Începînd cu Conversaţia 5, ne întîlnim cu astfel de situaţii pe tot parcursul lucrării. În schema logică din figura 5.7 este pus în evidenţă blocul de decizie cu o alternativă (selecţia simplă).

Se citeşte: **DACA** R(I) este mai mare ca zero **ATUNCI** mergi la linia 170.

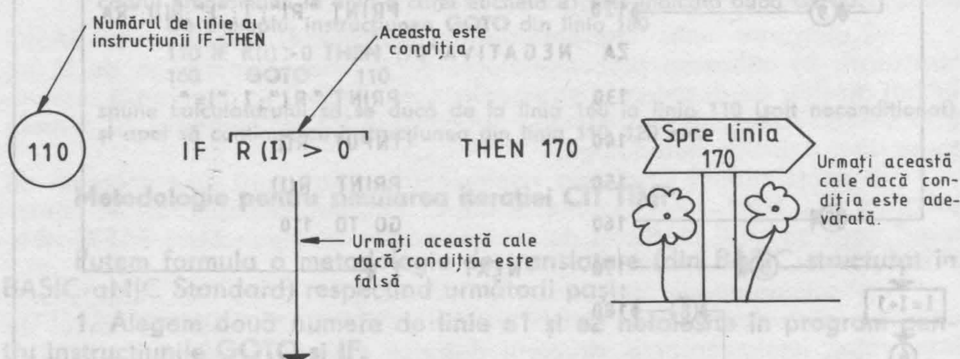
În BASIC-aMIC, o astfel de situaţie se scrie:

110 **IF** R (I)>0 **THEN** 170

Instrucţiunea **IF–THEN** este o instrucţiune de salt condiţionat. Formatul general al instrucţiunii este:

Format general
⟨nr. linie⟩ **IF** ⟨relaţie⟩ **THEN** ⟨nr. linie 1⟩

IF–THEN indică două continuări posibile, în funcţie de rezultatul evaluării condiţiei ⟨relaţie⟩ (o expresie relaţională). Dacă rezultatul evaluării expresiei relaţionale este de tipul "condiţie îndeplinită" atunci se execută instrucţiunea cu numărul de linie ⟨nr. linie 1⟩, iar dacă este de tipul ⟨condiţie neîndeplinită⟩ execuţia va continua cu instrucţiunea care urmează după IF.



Condiţia ce apare între cuvintele cheie **IF** şi **THEN** este de forma unei relaţii între două expresii numerice sau şiruri. Relaţiile se definesc cu ajutorul următorilor operatori relaţionali:

Simbol matematic	Simbol BASIC-aMIC	Semnificaţie	Tipuri operanzi
=	=	egal	Constante şi variabile numerice sau şir
<	<	mai mic	
⩽	<=	mai mic sau egal	
>	>	mai mare	
⩾	>=	mai mare sau egal	
≠	< >	diferit	

Aplicaţie. Identificaţi în BASIC-aMIC următoarele situaţii:

a) Dacă valoarea lui A este mai mică decît cinci atunci mergi la linia 80.
b) Dacă valoarea lui B este mai mare sau egală cu cea a lui C la pătrat atunci mergi la linia 30.
c) Dacă de 8 ori valoarea lui X nu este egală cu 7 atunci mergi la linia 800.

```
a) 10 IF A<5 THEN 80
b) 20 IF B>=C ↑2 THEN 30
c) 30 IF 8 * X <> 7 THEN 800
```

TEST

Se consideră următorul program BASIC:

```
10 FOR I=1 TO 9
20    READ A
30    IF A<0 THEN 50
40    PRINT "A="; A
50 NEXT I
60 DATA 4, 8, 0, -3, 6, -2, 7, 9, -4
70 END
```

a) Pentru care din valorile conţinute de blocul de date **DATA** condiţia A<0 este adevărată?
b) Cînd A<0 la ce linie se va face saltul?
c) Dacă condiţia A<0 este adevărată, valoarea lui A va mai fi afişată?
d) Pentru care din valorile conţinute de blocul de date **DATA** condiţia A<0 este falsă?
e) Cînd condiţia A<0 este falsă la ce linie se va face saltul?
f) Dacă condiţia A<0 este falsă, va mai fi afişată valoarea lui A?
g) Modificaţi instrucţiunea **IF-THEN** astfel încît programul să afişeze numai numerele diferite de zero.
h) Modificaţi instrucţiunea **IF-THEN** astfel încît programul să afişeze numai numerele mai mari sau egale cu 2.
i) Modificaţi instrucţiunea **IF-THEN** astfel încît programul să afişeze numai numerele: 7; 8; 9.

Instrucţiunea GOTO

În cadrul programului am utilizat instrucţiunea **GOTO** pentru construcţia unei bucle de tip **CIT TIMP.**

```
160 GO TO 110
```

Instrucţiunea **GOTO** defineşte un salt necondiţionat la orice instrucţiune din cadrul unui program.

Formatul instrucţiunii este:

Format general
⟨nr. linie⟩ **GOTO** ⟨nr. linie–n⟩

unde ⟨nr. linie–n⟩ reprezintă numărul de linie al instrucţiunii la care se face saltul (transferul).

Observaţie. Limbajul BASIC-aMIC, nu este un limbaj structurat. Deşi dăunătoare, instrucţiunea **GO TO** serveşte simulării structurilor algoritmice fundamentale.

TEST

Se consideră programul BASIC–aMIC

```
10 A=0
20 PRINT "A="; A
30 A=A+5
40 IF A>25 THEN 60
50 GOTO 20
60 PRINT "A>25"
70 END
```

a) Precizați liniile care definesc o buclă în cadrul programului.

b) De cîte ori instrucțiunea **IF-THEN** a verificat valoarea lui A înainte de a ști dacă condiția pusă a fost adevărată?

R. a) 20, 30, 40, 50; b) de șase ori.

Aplicație. Se consideră următorul program BASIC:

```
10 F=1                      40 F=F+1
20 IF F>8 THEN 99           50 GO TO 20
30 PRINT "F="; F            99 END
```

Modificați programul utilizînd o structură de iterație cu număr cunoscut de pași

```
10 FOR F=1 TO 8             30 NEXT F
20    PRINT "F="; F         40 END
```

□ Codificarea în limbajul BASIC-PRAE

vol. 2, pag. 210

Structura de selecție

În procesul de alcătuire a algoritmilor intră în sarcina noastră să exprimăm condițiile care urmează să fie evaluate de calculator în timpul execuției programului. Vom relua în cadrul acestui capitol problema structurilor de selecție.

Structura de selecție definește posibilitatea de selectare a unor structuri fundamentale algoritmice (de program), fiecare dintre ele avînd un singur punct de intrare și un singur punct de ieșire. Structura de selecție poate fi cu una (selecția simplă) sau cu două alternative (selecția).

În literatura de specialitate sînt precizate mai multe forme de reprezentare a structurilor de selecție.

Pentru selecția simplă (structura de decizie cu o singură alternativă) în cadrul acestei lucrări vom folosi reprezentarea

---

ETICHETĂ **DACA** ⟨condiție⟩

⟨GA_D – Grup de acțiuni pe ramura de DA⟩

ETICHETĂ **SFÎRȘIT**

Pentru selecție (structura de decizie cu două alternative) vom utiliza reprezentarea:

ETICHETĂ **DACA** ⟨condiție⟩

⟨GA_D – grup de acțiuni pe ramura de DA⟩

ETICHETĂ **ÎN CAZ CONTRAR**

⟨GA_N – grup de acțiuni pe ramura de NU⟩

ETICHETĂ **SFÎRȘIT**

Observaţii

- Instrucţiunile **DACA, IN CAZ CONTRAR** şi **SFÎRŞIT** au obligatoriu aceeaşi etichetă.
- Instrucţiunea **DACA** reprezintă punctul unic de intrare în bloc.
- Instrucţiunea **SFÎRŞIT** reprezintă punctul unic de ieşire din bloc.
- Dacă condiţia este adevărată se execută acţiunile cuprinse între **DACA** şi **IN CAZ CONTRAR** după care **IN CAZ CONTRAR** acţionează ca o instrucţiune **GOTO SFÎRŞIT.**
- Dacă condiţia este falsă se execută acţiunile cuprinse între **IN CAZ CONTRAR** şi **SFÎRŞIT.**

În figura 5.8 sînt prezentate sintetic formatele structurii alternative cu ajutorul schemelor logice, diagramei de structură şi pseudocodului.

Tip instrucţiune	Schema logică	Diagrama de structură	Pseudocod
DACA-ATUNCI	NU <condiţie> DA A	P A° ° Observaţie: Se citeşte: P constă din sau A. Cerculeţul semnifică "sau"(selecţia)	Etichetă DACA condiţie A Etichetă SFIRSIT
DACA-ATUNCI-IN CAZ CONTRAR	NU <condiţie> DA B A	P A° B° Observaţie: Se citeşte: P constă din sau A sau B.	Etichetă DACA condiţie A Etichetă IN CAZ CONTRAR B Etichetă SFIRSIT

Fig. 5.8. Reprezentarea structurilor alternative utilizînd schemele logice, diagrama de structură şi pseudocodul.

IF-THEN-ELSE

În BASIC-PRAE structurile de selecţie (selecţia şi selecţia simplă) se codifică cu instrucţiunea **IF-THEN-ELSE** (v. restricţiile instrucţiunii).

Formatul general este:

Format general
⟨nr. linie⟩ **IF** ⟨relaţie⟩ **THEN** ⟨nr. linie 1/instrucţiune 1⟩ [**ELSE** ⟨nr. linie 2/instrucţiune 2⟩]

în care:

⟨relaţie⟩ este de forma unei relaţii între două expresii (numerice sau şiruri) sau a unei reuniuni de relaţii (prin operatorii logici **AND, NOT, OR**);

⟨nr. linie 1⟩ poate fi orice număr de linie admis;

⟨instrucţiune 1⟩ poate fi orice instrucţiune executabilă exceptînd instrucţiunile: **FOR, NEXT, FNEND, END, DIM, DEF, IF, REM, DATA.**

Dacă rezultatul evaluării expresiei relaţionale este de tipul „condiţie îndeplinită", atunci se execută ⟨nr. linie 1/instrucţiune 1⟩, iar dacă este de tipul „condiţie neîndeplinită", atunci se execută ⟨nr. linie 2/instrucţiune 2⟩. În cazul cînd opţiunea **ELSE** lipseşte din instrucţiune atunci nu are loc nici un salt sau nu se execută ⟨instrucţiune 2⟩.

De notat că prin ⟨nr. linie 1/instrucţiune 1⟩ şi ⟨nr. linie 2/instrucţiune 2⟩ se înţelege fie saltul la o instrucţiune identificată prin numărul liniei de program, fie execuţia unei instrucţiuni (în afara celor exceptate), în funcţie de modul în care este scrisă instrucţiunea. În funcţie de evaluarea expresiei relaţionale se disting cazurile:

1. Relaţie adevărată

a) Dacă instrucţiunea este de forma:

IF ⟨relaţie⟩ **THEN** ⟨nr. linie 1⟩

atunci se execută instrucţiunea care are numărul de linie ⟨nr. linie 1⟩. De exemplu, instrucţiunea

20 **IF** A=8 **THEN** 80

are ca efect saltul la instrucţiunea din linia 80 şi execuţia acesteia.

b) Dacă instrucţiunea este de forma:

IF ⟨relaţie⟩ **THEN** ⟨instrucţiune 1⟩

atunci se execută instrucţiunea executabilă ⟨instrucţiune 1⟩ iar programul continuă cu următoarea instrucţiune executabilă (v. instrucţiunile exceptate).

De exemplu:

20 **IF** A>B **THEN** Y=5

are ca efect, atribuirea lui 5 variabilei Y (relaţie adevărată!).

Observaţie. În ambele situaţii opţiunea **ELSE** este ignorată.

2. Relaţie falsă

a) dacă opţiunea **ELSE** nu este specificată, atunci execuţia programului va continua cu următoarea instrucţiune executabilă;

b) dacă opţiunea **ELSE** este specificată, atunci continuarea execuţiei programului va avea loc în conformitate cu cazurile a şi b din situaţia precedentă.

De exemplu:

10 **IF** A=3 **THEN** 20 **ELSE** 180

are ca efect saltul la instrucţiunea din linia 180 şi execuţia acesteia (relaţie falsă!)

Observaţie. În ambele situaţii opţiunea **THEN** este ignorată.

Aplicaţie. Codificaţi în BASIC-PRAE următoarele situaţii:

a) Dacă valoarea lui A este mai mică decît 20 atunci mergi la linia 30.

b) Dacă valoarea lui C este mai mică sau egală cu cea a lui B ridicată la puterea a treia atunci afişează „EROARE".

c) Dacă de cinci ori valoarea lui A plus de trei ori valoarea lui B nu este egală cu 20 atunci afişează „EROARE", în caz contrar mergi la linia 200.

d) Dacă valoarea lui A este 2 atunci mergi la linia 800, în caz contrar mergi la linia 20.

---

a) 10 IF A<20 THEN 30
b) 20 IF C<=B ↑ 3 THEN PRINT "EROARE"
c) 30 IF(5*A+3*B)<>20 THEN PRINT "EROARE" ELSE 200
d) 40 IF A=2 THEN 800 ELSE 20

TEST

1) Se consideră următorul program:

```
10 FOR I=1 TO 9
20    READ A
30    IF A<0 THEN 50 ELSE PRINT "A="; A
40
50 NEXT I
60 DATA 1, 2, 0, -2, 5, -3, 8, 9, -8
70 END
```

a) Cînd A este mai mare sau egal cu zero, la ce linie se va face saltul?
b) Pentru care din valorile conţinute de blocul de date DATA se afişează mesajul "A="?
c) Dacă condiţia A<0 este adevărată va mai fi afişată valoarea lui A?
d) Modificaţi instrucţiunea **IF–THEN–ELSE** astfel încît programul să afişeze numai numerele diferite de zero?

2) Se consideră următorul program:

```
10 DIM A (20)
20 READ N
30 S=0
40 FOR I=1 TO N
50    READ A(I)
60    IF A(I)=0 THEN 80 ELSE
      S=S+A(I)
70 NEXT I
80 PRINT "S="; S
90 DATA 8, -6, 7, 0, 8, -3, -5, -2, -9
100 END
```

a) Care sînt funcţiunile programului?
b) De cîte ori se execută bucla **FOR-NEXT**?
c) Care este valoarea finală a variabilei S?

Tehnici de implementare a structurilor de selecţie. Metodologie

Codificarea selecţiei. Codificarea în BASIC structurat a selecţiei cu două ramuri este de forma **IF–THEN–ELSE.**

IF ⟨condiţie⟩ **THEN**		**DACĂ** ⟨condiţie⟩ **ATUNCI**
$\langle GI_D \rangle$		$\langle GI_D \rangle$
ELSE	în traducere:	**IN CAZ CONTRAR**
$\langle GI_N \rangle$		$\langle GI_N \rangle$
ENDIF		**SFIRŞIT DACĂ**

Codificarea structurii cu o ramură în BASIC structurat este de forma "IF—THEN".

IF ⟨condiţie⟩ **THEN**	**DACĂ** ⟨condiţie⟩ **ATUNCI**
⟨GI_D⟩	⟨GI_D⟩
ENDIF	**SFÎRŞIT DACĂ**

Simularea selecţiei

BASIC structurat → translatare → BASIC standard

IF ⟨condiţie⟩ **THEN**		**IF** ⟨complementul condiţiei⟩ **GOTO** e1
⟨GI_D⟩		⟨GI_D⟩
ELSE		**GOTO** e2
	e1	
⟨GI_N⟩		⟨GI_N⟩
ENDIF	e2	⟨instrucţiunea următoare⟩

Observaţie. Pentru translatare se foloseşte complementul condiţiei.

Simularea selecţiei simple

BASIC structurat → translatare → BASIC standard

IF ⟨condiţie⟩ **THEN**		**IF** (complementul condiţiei) **GOTO** e1
⟨GI_D⟩		⟨GI_D⟩
ENDIF	e1	⟨instrucţiunea următoare⟩

Metodologie de translatare

1. Alegem un număr de etichetă e1 nefolosit în program spre a fi utilizat în instrucţiunea **GO TO.**
2. Folosind complementul condiţiei, scriem instrucţiunea

   IF ⟨complementul condiţiei⟩ **GOTO** e1

3. Scriem instrucţiunile BASIC standard pentru grupul de instrucţiuni GI_D.
4. Pentru **selecţia simplă:** completaţi procesul de translatare plasînd imediat după ultima instrucţiune din GI_D.

   e1 ⟨instrucţiunea următoare⟩

   Pentru **selecţie** alegem un număr de etichetă nefolosit e2 şi adăugăm instrucţiunea **GOTO** e2 după ultima instrucţiune în GI_D. Ataşăm apoi numărul de linie e1 la prima instrucţiune în GI_N şi listăm instrucţiunile în GI_N.
5. Pentru **selecţie** adăugăm:

   e2 ⟨instrucţiunea următoare⟩

   imediat după ultima instrucţiune în GI_N.

Aplicaţie. Codificaţi în BASIC-PRAE următoarele situaţii:

a) Dacă valoarea lui A este mai mică decît 20 atunci: B, C, D şi E au valoarea 5.

b) Dacă valoarea lui A este mai mică decît 20 atunci: adună 5 la B, C, D, E iar în caz contrar adună 8.3 la X, Y, Z.

a)
```
10 IF A>=20 THEN 30
20 B=5 : C=5 : D=5 : E=5
30
```

b)
```
10 IF A>=20 THEN 40
20 B=B+5 : C=C+5 : D=D+5 : E=E+5
30 GO TO 50
40 X=X+8.3 : Y=Y+8.3 : Z=Z+8.3
50
```

Operatorii logici AND, OR

Într-o instrucţiune **IF–THEN–ELSE** se pot defini **condiţii compuse** utilizînd operatorii logici binari **AND, OR.** Se permite, de asemenea, utilizarea operatorului unar de complementare **NOT,** aplicat unei relaţii.

Reguli

- Dacă două relaţii sînt reunite prin operatorul logic **AND** (SI logic – corespunde simbolului matematic "X") perechea relaţiilor este adevărată numai dacă ambele relaţii sînt adevărate (v. figura 5.9).

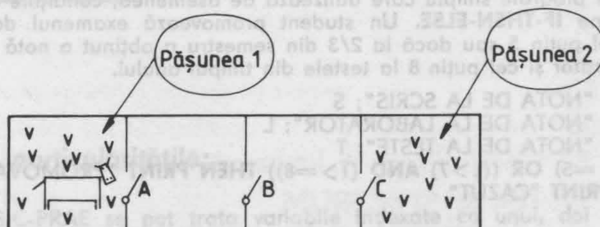


Fig. 5.9.

În figura 5.9, dacă poarta A şi B şi C sînt deschise, vaca Viorica poate să se ducă de pe păşunea 1 spre păşunea 2. Dacă una din porţi este închisă Viorica nu poate trece.

- Dacă două relaţii sînt reunite prin operatorul logic **OR** (SAU logic – corespunde simbolului matematic "+"), perechea relaţiilor este adevărată dacă cel puţin una dintre relaţii este adevărată. (v. figura 5.10).

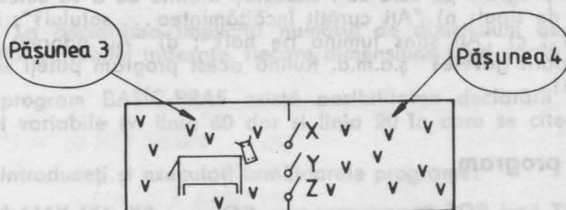


Fig. 5.10.

În figura 5.10, dacă poarta X sau Y sau Z sînt deschise, Viorica poate trece din pășunea 3 la pășunea 4.

În ciuda dificultăților întîlnite cu ocazia prezentării operatorilor logici, folosirea lor spre deosebire de notații este foarte simplă.

Aplicație. Următorul program ne ajută să înțelegem mai bine logica operatorului **AND.**

```
10 REM DA=1; NU=0
20 INPUT "POARTA A ESTE DESCHISĂ
           (1/0)"; A
30 INPUT "POARTA B ESTE DESCHISĂ
           (1/0)"; A
40 INPUT "POARTA C ESTE DESCHISĂ
           (1/0)"; A
50 PRINT
60 IF (A=1) AND (B=1) AND
   (C=1) THEN PRINT "VIORICA
   POATE PAȘTE PE PĂȘUNEA 2"
   ELSE PRINT "VIORICA RĂMÎNE
   PE PĂȘUNEA 1"
70 END
```

Liniile 20, 30 și 40 fixează poziția porții de intrare: deschis (1) închis (0).

Linia 60 reprezintă cheia programului. Dacă toate porțile sînt deschise „VIORICA POATE PAȘTE PE PĂȘUNEA 2". Dacă oricare din porți este închisă "VIORICA RĂMÎNE PE PĂȘUNEA 1".

TEST

Folosind programul de mai sus ca model scrieți un altul care să sugereze logica operatorului **OR** (de preferat ca personajul principal să rămînă tot VIORICA).

60 **IF** (A=1) **OR** (B=1) **OR** (C=1) **THEN** PRINT "POARTA ESTE DESCHISĂ. VIORICA POATE TRECE" **ELSE PRINT** "VIORICA RĂMÎNE PE PĂȘUNEA 1".

Aplicații

a) Iată un program simplu care utilizează de asemenea, condițiile compuse într-o instrucțiune **IF-THEN-ELSE.** Un student promovează examenul dacă a obținut la scris cel puțin 5 sau dacă la 2/3 din semestru a obținut o notă mai mare decît 7 la laborator și cel puțin 8 la testele din timpul anului.

```
10 INPUT "NOTA DE LA SCRIS"; S
20 INPUT "NOTA DE LA LABORATOR"; L
30 INPUT "NOTA DE LA TESTE"; T
40 IF (S>=5) OR ((L>7) AND (T>=8)) THEN PRINT "PROMOVAT"
   ELSE PRINT "CĂZUT"
50 END
```

b) Următorul program studiază dacă două numere (citite dinamic) sînt ambele pozitive, negative sau de semne contrare.

```
10 INPUT "PRIMUL NUMĂR ESTE"; A
20 INPUT "AL DOILEA NUMĂR ESTE"; B
30 IF (A=0) AND (B=0) THEN
   PRINT "AMBELE POZITIVE"
40 IF (A<0) AND (B<0) PRINT "AMBELE NEGATIVE"
   ELSE PRINT "SEMNE DIFERITE"
50 END
```

c) Scrieți un program pe care să-l executați înainte de a vă culca. Se vor programa întrebări de tipul: a) "Ați curățit încălțămintea ... soțului?"; b) "Ați pus ceasul să sune?"; c) "Ați stins lumina pe hol?"; d) "Au adormit cei patru copii?"; e) "Ați oprit gazele?" ș.a.m.d. Rulînd acest program puteți scăpa de eventuale coșmaruri!

Înapoi la program

Ne-am luat cu vorba și n-am spus nimic în legătură cu variabilele indexate din cadrul programului.

```
20 INPUT "NUMAR REZERVOARE";N
40 PRINT N
50 REM CITIRE (DINAMICA) SI V
ALIDARE DATE
60 DIM R(N)
70 FOR I=1 TO N
80   PRINT "R(";I;")=";
90   INPUT R(I)
100  PRINT R(I)
110  IF R(I)>0 THEN 170
120  PRINT "R(";I;")=";R(I);
"RAZA NEGATIVA"
130  PRINT "R(";I;")=";
140  INPUT R(I)
150  PRINT R(I)
160  GO TO 110
170 NEXT I
180 REM CITIRE (STATICA) DENS
ITATE BENZINA "
190 READ D
200 REM CALCULUL MASEI DE BEN
ZINA DIN REZERVOARE
210 PRINT
220 REM TIPARIRE CAP TABEL
230 PRINT "RAZA"," MASA"
232 FOR I=1 TO 28
234   PRINT "=";
236 NEXT I
240 S=0
250 FOR I=1 TO N
260   M=D*2*PI*R(I)^3
270   S=S+M
280   PRINT R(I),M
290 NEXT I
300 FOR I=1 TO 28
310   PRINT "-";
320 NEXT I
330 REM TIPARIRE REZULTAT FIN
AL
340 PRINT "MASA TOTALA=",S;"T
ONE"
350 DATA .7
360 END
```

Iată, pe scurt, **particularităţile:**

- În BASIC-PRAE se pot trata variabile indexate cu unul, doi sau pînă la 255 indici.
- Numele unei variabile indexate este alcătuit dintr-un nume admis de variabila simplă urmat de una sau două expresii de indici în paranteze.
- Într-un program BASIC-PRAE este posibilă definirea prin acelaşi nume a unei variabile simple şi a unei variabile indexate.
- Într-un program BASIC-PRAE declararea tablourilor se poate realiza în două moduri:

– declarare explicită prin instrucţiunile **DIM** (vezi linia 60) şi **COM**;

– declararea implicită prin referirea la un element al tabloului (v. liniile 90, 100, 110, 260, 280).

Observaţie. La declararea implicită numărul de dimensiuni al tabloului este egal cu numărul indicilor variabilei indexate, fiecare dimensiune fiind egală cu 10.

- Într-un program BASIC-PRAE există posibilitatea declarării unui tablou cu dimensiuni variabile (v. linia 60 dar şi linia 20 în care se citeşte valoarea lui N).

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 REM MAX (X1, X2, ..., XN)
20 READ N
30 DIM X(N)
40 FOR I=1 TO N
50       READ X(I)
60 NEXT I
```

```
70 DATA 10
80 DATA 10, 5, -8, 0.9,
   -28, 6, 7, 8, 9, -4
90 M=X(1)
100 FOR I=2 TO N
110      IF M>=X(I) THEN 130
120      M=X(I)
130 NEXT I
140 PRINT "MAX="; M
150 END
```

Remarcă. Încercaţi să realizaţi singuri, fără ajutorul nostru, tabela de variabile şi schema logică ale acestui program.

b)

```
5 FOR I=1 TO 5
10 INPUT "INTRODUCEŢI UN
   NUMĂR"; A
20 PRINT
30 IF A>1988 THEN PRINT "A>1988":
   GOTO 70
40 IF A=1988 THEN PRINT
   "A=1988": GOTO 70
50 PRINT "A<1988"
60 PRINT
70 CLS
80 NEXT I
```

c)

```
10 CLS
20 PRINT "INTRODUCEŢI ORA
   EXACTĂ: ORA, MINUTUL,
   SECUNDA"
30 PRINT "EXEMPLU : 10, 20, 38"
40 INPUT H, M, S
50 CLS
60 C=0
70 PRINT H; " : "; M; " : "; S
80 C=C+1
90 IF C<100 THEN 80
100 S=S+1
110 IF S<60 THEN 50
120 S=0
130 M=M+1
140 IF M<60 THEN 50
150 M=0
160 H=H+1
170 IF H<=12 THEN 50
180 H=1
190 GO TO 50
200 END
```

□ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 211

Variabile indexate numerice. Reguli

În limbajul BASIC HC-85, TIM S, SPECTRUM variabilele indexate prezintă următoarele **particularităţi:**

- numele unei variabile indexate este alcătuit dintr-o singură literă;
- tablourile numerice sînt memorate pe un număr de 4+2 ∗ (numărul dimensiunilor) + 5 ∗ (număr de elemente) octeţi;
- o variabilă numerică simplă şi o variabilă tablou (indexată) pot avea acelaşi nume deoarece sînt distincte ca tip.

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 DIM A(4)
20 FOR I=1 TO 4
40    READ A(I)
50 NEXT 1
60 FOR I=1 TO 4
70    PRINT TAB 6; "A ("; I; ")="; A(I)
80 NEXT I
90 DATA 6, 9, 7, 1
100 END
```

Înapoi la program

Programul utilizează o singură variabilă indexată de tip numeric R(I).

```
90 INPUT R(I)
100 PRINT R(I)
110 IF R(I)>0 THEN GOTO 140
120 PRINT "R ("; I; ")="; R(I); "R
    AZA NEGATIVĂ"
.
.
.
230 LET M=D*2*PI*R(I)↑3
.
.
250 PRINT R(I), M
```

Remarcaţi modul în care s-a folosit această variabilă în cele şase linii de program de mai sus.

DIM

În limbajul BASIC HC-85, TIM S, SPECTRUM instrucţiunea **DIM** prezintă următoarele **particularităţi:**

- iniţializează elementele tabloului cu zero sau blanc (variabile tablou/şir);
- şterge orice tablou care are acelaşi nume cu variabila definită prin instrucţiunea curentă;
- cu o instrucţiune **DIM** poate fi definită numai o singură variabilă indexată (de tip tablou);
- există posibilitatea declarării unui tablou cu dimensiuni variabile.

Aplicaţie. Să se scrie un program BASIC care calculează valoarea polinomului:

$$5x^2+4x+10$$

pentru x=2.

Indicaţie. Se va utiliza schema lui Horner:

```
10 REM * SCHEMA LUI HORNER *
20 READ N, X
30 DIM A(N)
40 FOR I=1 TO N
50    READ A(I)
60 NEXT I
70 P=A(N)
80 FOR I=N-1 TO 1 STEP -1
90    LET P=P*X+A(I)
100 NEXT I
110 PRINT "VALOAREA POLINOMULUI="; P
120 DATA 3, 2, 10, 4, 5
130 END
```

Remarcă. Coeficienţii 10, 4 şi 5 se află în vectorul A după cum urmează: 10 în A(1), 4 în A(2) şi 5 în A(3). N reprezintă numărul de termeni iar X gradul polinomului. P este valoarea polinomului, ce se iniţializează cu 5.

Înapoi la program

Programul utilizează instrucţiunea **DIM** pentru rezervarea spaţiului de memorie corespunzător variabilei indexate R(I).

```
30 INPUT N
40 PRINT N
.
.
60 DIM R(N)
.
.
.
```

Remarcaţi în linia 60 a programului, modul de declarare a tabloului R – cu dimensiuni variabile (N), valoarea lui N fiind citită anterior, în linia 30.

IF-THEN. Reguli

Instrucţiunea BASIC care realizează luarea deciziilor este de forma **IF-THEN.** Formatul general al instrucţiunii este:

Format general
⟨nr. linie⟩ **IF** ⟨relaţie⟩ **THEN** ⟨acţiune⟩

în care:

⟨relaţie⟩ este de forma unei relaţii între două expresii (numerice sau şiruri) sau a unei reuniuni de relaţii prin operatorii logici: **AND, NOT, OR;** ⟨acţiune⟩ poate fi o instrucţiune **GOTO,** o altă instrucţiune sau o secvenţă de instrucţiuni.

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 INPUT "Introduceţi un număr
   între 1 şi 3"; a
20 IF a=1 THEN PRINT "unu"
30 IF a=2 THEN PRINT "doi"
40 IF a=3 THEN PRINT "trei"
50 GO TO 10
```

b)
```
10 INPUT x
20 INPUT y
30 IF x=1 AND y=1 THEN PRINT
   "adevărat"
40 GO TO 10
```

c)
```
10 INPUT a
20 INPUT b
30 IF a=1 THEN IF b=2 PRI
   NT "adevărat"
```

d)
```
10 INPUT a
20 INPUT b
30 IF a=1 AND b=2 THEN PRINT
   "adevărat"
```

e)
```
10 INPUT m
20 IF m THEN PRINT m
```

f)
```
10 INPUT m
20 IF NOT m THEN PRINT m
```

g)
```
10 LET a=0
20 PRINT "BASIC"
30 LET a=a+1
40 IF a<25 THEN GO TO 20
```

h)
```
10 LET a=0
20 LET a=a+1
30 IF a<25 THEN PRINT "BASIC"
40 GO TO 20
```

TEST

Să se codifice în BASIC HC-85, TIM S, SPECTRUM următoarea situaţie: dacă x=1 atunci y=1 altfel, dacă x=5 mergi la linia 100.

Varianta 1

```
10 IF X=1 THEN LET Y=1
20 IF X< >1 THEN IF X=5 THEN GO TO 100
```

Varianta 2

```
10 IF X=1 THEN LET Y=1
20 IF X< >1 AND X=5 THEN GOTO 100
```

Expresii condiţionale

În BASIC HC-85, TIM S, SPECTRUM operatorii logici **(AND, OR, NOT)** pot fi consideraţi şi folosiţi ca funcţii:

OPERATOR	Expresie condiţională	Rezultat
AND	a AND b	a dacă b < > 0
		0 dacă b = 0
OR	a OR b	1 dacă b < > 0
		a dacă b = 0
NOT	NOT a	0 dacă a < > 0
		1 dacă a = 0

Aplicaţii

a) Se consideră programul

```
10 LET a=0
20 LET b=a
30 LET a=a+1
40 IF a>31 THEN LET a=0
50 PRINT "b=" ; b
60 GO TO 20
```

Putem economisi o linie de program modificînd linia 30.

```
30 LET a=a+1 AND a<31
```

Modul în care lucrează linia 30 este puţin mai complex. Dacă valoarea lui a este inferioară lui 31 atunci adaugă 1 la valoarea lui a dar dacă valoarea lui a nu este inferioară lui 31 atunci iniţializează-l pe a cu zero. Ştergeţi acum linia 40 (pentru a nu avea redundanţe) şi renumerotaţi din 10 în 10 instrucţiunile programului. Vom obţine:

```
10 LET a=0
20 LET b=a
30 LET a=a+1 AND a<31
40 PRINT "b=" ; b
50 GO TO 20
```

b) Se consideră programul

```
10 LET x=15
20 IF b=5 AND x>1 THEN
   LET x=x-2
30 IF b=8 AND x<30 THEN
   LET x=x+2
40 PRINT "x=" ; x
50 GO TO 20
```

Puteţi obţine acelaşi rezultat (x=) cu:

```
10 LET x=15
20 LET x=x-(b=5 AND x>1) *
   2+(b=8 AND x<30) * 2
40 PRINT "x=" ; x
50 GO TO 20
```

sau cu:

```
10 LET x=15
20 LET x=x-(2 AND b=5 AND
   x>1)+(2 AND b=8 AND x<30)
40 PRINT "x="; x
50 GO TO 20
```

Dacă doriţi, puteţi citi ultima linie 20 şi cu voce tare: atribuie lui x valoarea: x-(2, dacă b=5 şi valoarea lui x este superioară lui 1, în caz contrar 0)+(2 dacă b=8 şi valoarea lui x este inferioară lui 30, în caz contrar 0).

Remarcă. Valorile ADEVĂRAT şi FALS sînt reprezentate respectiv prin 1 şi zero.

```
10 INPUT a
20 INPUT b
30 LET x=(a=b)
40 PRINT "x="; x
50 GO TO 10
```

Parantezele din linia 30 nu sînt absolut necesare dar ele permit o înţelegere mai uşoară a sensului expresiei. Expresia dintre paranteze nu ia decît valoarea 1 sau 0 atunci cînd a este sau nu este egal cu b.

TEST

Precizaţi valoarea lui x.

```
10 INPUT a
20 INPUT b
30 LET x=(a=b) * 3
40 PRINT "x="; x
50 GOTO 10
```

De această dată veţi obţine o valoare x=0 pentru a≠b şi o valoare x=3 atunci cînd a=b.

Înapoi la program

Programul utilizează o singură dată instrucţiunea **IF**, în linia 110. Remarcaţi între **IF** şi **THEN** condiţia R(I)>0

```
110 IF R(I)>0 THEN GOTO 140
```

iar după **THEN** instrucţiunea de salt necondiţionat **GOTO**.

TEST

Rescrieţi secvenţa de instrucţiuni din liniile 70–140 ale programului astfel încît pentru procesul de validare date, să realizaţi o structură de iteraţie de tip **CÎT TIMP**.

```
70 FOR I=1 TO N
80    PRINT "R(" ; I; ")=" ;
90    INPUT R (I)
100   PRINT R (I)
110   IF R (I)>0 THEN GO TO 140
120   PRINT "R ("; I; ")=" ; R (I); "RAZA NEGATIVĂ"
125   PRINT "R ("; I; ")=";
130   INPUT R (I)
135   GO TO 110
140 NEXT I
```

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 FOR I=1 TO 10
20    IF I=9 THEN GO TO 40
30 NEXT I
40 PRINT "ÎN LINIA 40, I="; I
50 FOR I=1 TO 2
60    FOR J=1 TO 10
70          PRINT I; J,
80       NEXT J
90 NEXT I
100 END
```

b)
```
10 REM MAX (X1, X2, . . . , XN)
20 REM MIN (X1, X2, . . . , XN)
30 READ N
40 DIM X(N)
50 FOR I=1 TO N
60    READ X(I)
70 NEXT I
80 DATA 10
90 DATA 3, -8, 0, 7, 15, -25, -30, 8, 9, 10
100 LET M1=X(1)
110 LET L=1
```

```
120 LET M2=X(1)
130 LET K=1
140 FOR I=2 TO N
150     IF X(I) >M1 THEN GOTO 220
160     IF X(I) >=M2 THEN GOTO 220
170     LET M2=X(I)
180     LET K=I
190     GOTO 220
200     LET M1=X(I)
210     LET L=I
220 NEXT I
230 PRINT "MAX"; M1; "POZIŢIA"; L, "MIN="; M2; "POZIŢIA"; K
240 END
```

Remarcă. Încercaţi să realizaţi singuri, fără ajutorul nostru, specificaţiile de programare şi documentaţia de proiectare ale acestui program.

c)

```
10 PRINT "INTRODUCEŢI UN NUMĂR ÎNTREG";
20 INPUT N
30 LET S=0
40 FOR I=1 TO N
50   LET S=S+(N/I=INT (N/I))
60 NEXT I
70 PRINT "S="; S
80 END
RUN
INTRODUCEŢI UN NUMĂR ÎNTREG ? 1949
S=2
```

d)

```
10 FOR I=1 TO 20
20      PRINT I; 20/I; INT (20/I); 20/I–INT (20/I)
30 NEXT I
40 END
```

Remarcă. Funcţia **INT** (x) furnizează partea întreagă a lui x. Formatul general este:

Format general
INT (⟨expresie numerică⟩)

□ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 212

Variabile indexate (numerice). Reguli

În limbajul BASIC-AMSTRAD variabilele indexate (numerice) prezintă următoarele **particularităţi:**

- numele unei variabile indexate este alcătuit dintr-un nume admis de variabilă simplă urmat de un ansamblu de indici cuprins între paranteze; indicii pot fi: nume de variabile, expresii, funcţii care prin evaluare trebuie să dea o valoare întreagă pozitivă;
- este posibilă definirea prin acelaşi nume a unei variabile simple şi a unei variabile indexate;
- valoarea minimă a unui indice este zero (primul element al unui tablou);
- variabilele indexate (numerice) pot fi de două tipuri: întreg, real. Tipul întreg se indică prin caracterul "%" plasat după numele simbolic al variabilei.

În cadrul programului variabila indexată numerică cu un indice este R(I).

DIM

În limbajul BASIC-AMSTRAD instrucţiunea DIM prezintă următoarele **particularităţi:**

- declararea explicită a unui tablou (prin **DIM**) trebuie plasată, în program înainte de prima referire a unui element al tabloului;
- dacă dimensiunea (**DIM**) unei variabile indexate nu este specificată atunci ea este stabilită implicit la 10;
- instrucţiunea **DIM** stabileşte ca valoare iniţială zero a tuturor variabilelor din tablou;
- cînd un tablou nu mai este necesar el poate fi şters (**ERASE**) cu instrucţiunea **ERASE,** al cărei format este

Format general
⟨număr linie⟩ **ERASE** ⟨listă variabile⟩

De exemplu

```
10 DIM b (100)
20 ERASE b
```

are ca efect ştergerea din memorie a vectorului b în vederea utilizării acesteia pentru alte scopuri;
- într-un program BASIC-AMSTRAD există posibilitatea declarării unui tablou cu dimensiuni variabile.

În cadrul programului s-a declarat vectorul R

```
120 DIM R (N)
```

cu instrucţiunea **DIM** de dimensiuni variabile (valoarea lui N se citeşte în linia 90 a programului).

Aplicaţie. Se consideră vectorii:

a=[1 2 3]

şi

b=[4 5 6]

Să se scrie un program BASIC care calculează produsul vectorial

a x b = [−3, 6, −3]

```
5 REM PRODUSUL VECTORIAL A DOI VECTORI
10 FOR I=1 TO 3
20        READ A(I), B(I)
30 NEXT I
40 DATA 1, 4, 2, 5, 3, 6
50 LET C(1)=A(2) * B(3)-B(2) * A(3)
60 LET C(2)=-A(1) * B(3)+B(1) * A(3)
70 LET C(3)=A(1) * B(2)-B(1) * A(2)
80 FOR I=1 TO 3
90        PRINT C(I);
100 NEXT I
110 END
```

IF–THEN–ELSE

În limbajul BASIC-AMSTRAD instrucţiunea **IF–THEN–ELSE** prezintă următoarele **particularităţi:**

- atunci cînd rezultatul condiţiei (plasate între **IF** şi **THEN**) necesită un salt de linie, instrucţiunea se poate formula ca în exemplul de mai jos:

10 **IF** x=1 **THEN** 200

sau:

10 **IF** x=1 **GOTO** 200

sau:

10 **IF** x=1 **THEN GO TO** 200

- ⟨relaţie⟩ (v. formatul instrucţiunii din limbajul BASIC-PRAE) este de forma unei relaţii între două expresii (numerice sau şiruri) sau a unei reuniuni de relaţii prin operatorii logici: **AND, NOT, OR, XOR** (sau exclusiv).

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 INPUT "Astăzi sîntem în"; ziua
20 INPUT "Precizaţi numărul lunii"; luna
30 IF Ziua=1 AND luna=9 THEN 50
40 CLS: GOTO 10
50 PRINT "La mulţi ani pentru 1 septembrie"
```

b)
```
10 CLS
20 INPUT "Precizaţi numărul lunii"; luna
30 IF luna=12 OR luna=1 OR luna=2 THEN 50
40 GO TO 10
50 PRINT "Sîntem în anotimpul iarna!"
```

c)
```
10 CLS
20 INPUT "Precizaţi numărul lunii"; luna
30 IF NOT (luna=6 OR luna=7 OR luna=8) THEN 50
40 GOTO 10
50 PRINT "Nu sîntem în anotimpul vara!"
```

d)
```
10 INPUT "Astăzi sîntem în"; ziua
20 INPUT "Precizaţi numărul lunii"; luna
30 IF NOT (luna=12 OR luna=1) AND ziua=29 THEN 50
40 CLS : GOTO 10
50 PRINT "Nu este nici decembrie nici ianuarie dar poate fi un an bisect"
```

TEST

Care este rezultatul execuţiei următoarelor instrucţiuni

a) 10 **PRINT** 10 **AND** 10

R . 10

b) 10 **PRINT** 10 **AND** 12

R . 8

c) 10 **PRINT** 10 **AND** 1000

R . 8

d) 10 **PRINT** 1000 **OR** 10

R . 1002

Observaţie. **XOR** (sau exclusiv) dă un rezultat adevărat dacă cele două argumente sînt diferite (vezi tabela de adevăr).

ARGUMENTE		OPERATOR		
A	B	**AND**	**OR**	**XOR**
1010	0110	0010	1110	1100

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 CLS
20 A=0
30 PRINT "ZERO"
40 INPUT C$
50 CLS
60 A=A+1
70 IF A=1 THEN PRINT "UNU"
80 IF A=2 THEN PRINT "DOI"
90 IF A=3 THEN PRINT "TREI"
100 IF A=4 THEN PRINT "PATRU"
110 IF A=5 THEN PRINT "CINCI"
120 IF A<=5 THEN 40
130 GO TO 10
140 END
```

Remarci

- instrucţiunea **INPUT** din linia 40 are o importanţă deosebită pentru buna desfăşurare a lucrurilor din cadrul acestui program. Fără această instrucţiune, calculatorul ar fi afişat mesajele din liniile 70/80/90/100/110 doar pentru o fracţiune de secundă, căci urmare a execuţiei instrucţiunilor din liniile 120 şi 130 ecranul s-ar fi şters imediat. Aceasta demonstrează valoarea inserării instrucţiunilor de tipul

  40 **INPUT** C$

  şi a răspunde la aceste instrucţiuni cu o valoare nulă – prin simpla apăsare pe tasta [**RETURN**] numai după ce aţi avut timp să vedeţi rezultatele afişate pe ecran.
- Folosirea instrucţiunii

  130 **GO TO** 10

  dă naştere la o **buclă infinită**, programul neavînd posibilitatea să se oprească de la sine. De fapt, există o instrucţiune **END** în linia 140 dar cum ea este precedată de instrucţiunea **GOTO** înseamnă că instrucţiunea **END** nu va fi niciodată executată. În acest fel, programul nu se sfîrşeşte niciodată iar calculatorul va lucra pînă la scoaterea lui din funcţiune.

Ei bine, de fapt există o cale de ieşire din această buclă cauzată de instrucţiunea **GOTO** şi anume acţionarea tastei [**ESC**]. Deci, o cale de ieşire dintr-o buclă o reprezintă apăsarea tastei [**ESC**]. După aceasta, puteţi modifica programul, puteţi introduce instrucţiuni **PRINT** pentru a lista valorile variabilelor, puteţi scrie un nou program ori să-l reluaţi pe cel vechi introducînd o comandă **RUN.**

TEST

Precizaţi rezultatul execuţiei următoarelor instrucţiuni BASIC:

a) `10 PRINT "<=>"; : GO TO 10`
b) `10 PRINT "<=>" : GO TO 10`
c) `10 PRINT "<=>", : GO TO 10`
d) `10 PRINT TAB (16); "<=>" : GO TO 10`

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 FOR I=1 TO 6
20 INPUT "A, B"; A, B
30 IF A>B THEN PRINT "PRIMUL NUMĂR"; A; "ESTE MAI MARE DECÎT AL
   DOILEA" : PRINT "NUMĂR"; B : GOTO 70
40 IF A=B THEN PRINT "PRIMUL NUMĂR"; A; "ESTE EGAL CU AL DOILEA"
   : PRINT "NUMĂR"; B : GOTO 70
50 PRINT "PRIMUL NUMĂR"; A; "ESTE MAI MIC DECÎT AL DOILEA" :
   PRINT "NUMĂR"; B
70 NEXT I
```

b)
```
5 FOR I=1 TO 4
10 INPUT "A, B, C" ; A, B, C
20 IF A*B>A+B*C THEN PRINT A; " * "; B; " > "; A; " + "; B; " * ";
   C : GO TO 60
30 IF A*B=A+B*C THEN PRINT A; " * "; B; " = "; A; " + "; B; " * ";
   C : GO TO 60
40 PRINT A; " * "; B; " < "; A; " + "; B; " * "; C
60 NEXT I
```

c)
```
10 INPUT "A, B"; A, B
20 IF A>B THEN PRINT "A>" : PRINT "B" : GOTO 60
30 IF A=B THEN PRINT "A=" : PRINT "B" : GO TO 60
40 PRINT "A<": PRINT "B"
50 PRINT
60 PRINT "TASTATI 1 PENTRU CONTINUARE"
70 PRINT "TASTATI 0 PENTRU OPRIRE"
80 INPUT N
90 IF N=1 THEN 10
100 IF N<>0 THEN 80
110 PRINT
120 END
```

d)
```
10 REM -32 767 şi +32 767
20 REM - - - - - - - - - -
30 AMINUS % = -32760
40 APLUS % = +32760
50 PRINT "LOW VALUE", "HIGH VALUE"
60 AMINUS % = AMINUS %-1
70 APLUS % = APLUS % + 1
80 PRINT AMINUS %, APLUS %
90 GO TO 60
```

e)
```
10 CLS
20 FOR I=0 TO 13
30 PRINT TAB (I), "<=>"
40 NEXT
50 GO TO 50
```

Observație. Remarcați efectul instrucțiunii

```
50 GO TO 50
```

Programul ciclează, ecranul rămînînd tot timpul cu aceeaşi imagine – 14 vaporaşe. Pentru ieşirea din buclă tastaţi [**ESC**].

WHILE–WEND

Instrucțiunea creează un ciclu alcătuit din mai multe instrucțiuni, atîta timp cît o condiţie dată este adevărată.

Formatul general al instrucțiunii este

Format general
WHILE
⟨secvență – instrucțiuni⟩
WEND

în care:

⟨relaţie⟩ are semnificația cunoscută;

⟨secvență – instrucțiuni⟩ reprezintă orice tip de instrucţiune BASIC-AMSTRAD.

Remarci

- **WHILE-WEND** implementează în BASIC structurat structura de iterație **CÎT-TIMP**.
- **WHILE** indică punctul de intrare în ciclu.
- **WEND** precizează punctul de ieșire din ciclu.
- **WHILE-WEND** poate fi executat cel puțin de zero ori.
- Se pot folosi cicluri **WHILE-WEND** imbricate în care fiecare **WEND** închide pe cel mai apropiat **WHILE**.

Joc pe calculator

Introduceți și executați următorul program:

```
10 READ A
15 DATA 7
20 INPUT "INTRODUCEȚI UN NUMĂR (0 LA 9)"; X
30 PRINT "AȚI INTRODUS NUMĂRUL : "; X
40 WHILE A< >X
50      IF X<A THEN PRINT "NUMĂRUL"; X; "ESTE PREA MIC! INTRODUCEȚI
        ALTUL" : GOTO 70
60      IF X>A TEHN PRINT "NUMĂRUL"; X; "ESTE PREA MARE! INTRODU-
        CEȚI ALTUL"
70      INPUT "INTRODUCEȚI UN NUMĂR (0 LA 9)"; X
80      PRINT "AȚI INTRODUS NUMĂRUL : "; X
90 WEND
100 PRINT "FELICITĂRI! NUMĂRUL ESTE : "; A
110 END
```

Aplicație. Să se scrie un program care calculează sin (x). Se va utiliza formula:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!}$$

```
10 INPUT "X="; X
20 INPUT "E="; E
30 S=0
40 N=0
50 N1=1
60 T=(-1) ↑ N * X ↑ (2 * N+1)/N1
70 WHILE ABS (T) >=E
80      S=S+T
90      N=N+1
100     N1=N1 * 2 * N * (2 * N+1)
110     T=(-1) ↑ N * X ↑ (2 * N+1)/N1
120 WEND
130 PRINT "X="; X, "SIN (X)=", SIN (X), "S="; S
140 END
RUN
X=? 1
E=? 0.001
X=1 SIN (X)= ·84147098 S= ·84166667
```

Remarci

- **SIN** (X) este funcția aritmetică sinus de x. Are ca rezultat sin (x) calculat în simplă precizie, unde x este exprimat în radiani.
  Formatul general este:

Format general
SIN (⟨expresie numerică⟩)

- **ABS** (X) dă valoarea absolută a lui X sau a expresiei numerice cuprinse între paranteze.

Formatul general este:

Format general
ABS (⟨expresie numerică⟩)

Înapoi la program

În program instrucţiunea **WHILE–WEND** creează un ciclu compus din instrucţiunile:

```
170 WHILE R(I)<0
180     PRINT "R("; I; ")="; R(I);
190     PRINT "R("; I; ")=";
200     INPUT R(I)
210     PRINT R(I)
220 WEND
```

De notat că bucla **WHILE–WEND** este inclusă în bucla **FOR–NEXT.** Remarcaţi, de asemenea, că în interiorul buclei **WHILE** valoarea variabilei I rămîne aceeaşi cît timp R(I) are o valoare negativă. La ieşirea din buclă, se dă controlul instrucţiunii **FOR–NEXT.**

Aplicaţie: Introduceţi şi executaţi următorul program:

```
10 PRINT "INTRODUCEŢI UN NU-
   MĂR ÎNTREG (1–6 CIFRE)";
20 INPUT N
30 WHILE N<>999999
40   FOR I=5 TO 0 STEP –1
50       E=INT (N/10↑I)
60       PRINT E
70       R=N–E * 10↑I
80       N=R
90   NEXT I
100  PRINT "INTRODUCEŢI UN NU-
     MĂR ÎNTREG (1–6 CIFRE)";
110      INPUT N
120 WEND
RUN
INTRODUCEŢI UN NUMĂR ÎNTREG (1–6
   CIFRE)? 76321
   7
   6
   3
   2
   1
INTRODUCEŢI UN NUMĂR ÎNTREG (1–6
   CIFRE)? 999999
```

Joc pe calculator

Gîndiţi-vă la două numere (A şi B) mai mici decît 10. Înmulţiţi pe A cu 2 şi adunaţi 5 la produsul obţinut. Înmulţiţi rezultatul obţinut cu 5. Adunaţi la noul rezultat pe 8. Spuneţi cît aţi obţinut şi vă vom spune la ce numere v-aţi gîndit.

Indicaţie. Numărul obţinut este de forma 10A+B=A zeci+B unităţi. Pentru determinarea lui A şi B consultaţi aplicaţia precedentă.

TEST

Precizaţi care sînt rezultatele execuţiei următorului program. Introduceţi şi executaţi programul pe calculatorul dvs.

```
10 N=0
20 S=0
30 READ D
40 WHILE D<>999
50       N=N+1
60       S=S+D
80 WEND
90 S=S/N
100 PRINT "MEDIA ARITMETICĂ="; S
110
120 REM BLOC DATE
130 DATA 98, 80, 73, 92, 77, 84,
    83, 79, 87, 73
140 DATA 99, 63, 63, 92, 81, 93,
    47, 53, 89, 100
150 DATA 98, 71, 73, 999
160 END
RUN
```

□ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 213

Variabile indexate (numerice). Reguli

În limbajul BASIC-COMMODORE variabilele indexate (numerice) prezintă următoarele **particularităţi:**

- numele unei variabile indexate este alcătuit dintr-un nume admis de variabila simplă urmat de unu, doi sau trei indici cuprinşi între paranteze;
- indicii pot fi: nume de variabile, expresii, funcţii;
- pot exista în cadrul aceluiaşi program variabile simple şi variabile indexate cu acelaşi nume;
- valoarea minimă a unui indice este zero;
- variabilele indexate pot fi de două tipuri: întreg, real. Tipul întreg se indică prin caracterul "%" plasat după numele simbolic al variabilei.

Revenind la program, semnalăm utilizarea de către program a variabilei indexate numerice, cu o dimensiune R(I).

DIM

Instrucţiunea **DIM** spre deosebire de cea din BASIC-AMSTRAD prezintă următoarea particularitate: atunci cînd dimensiunea unei variabile indexate nu este specificată atunci ea este stabilită implicit la 11.

Aplicaţii.

a) Se consideră vectorii:

a=[6 2 1]

şi

b=[1 2 5].

Să se scrie un program BASIC care calculează

a+b=[7 4 6]

```
5 REM ADUNAREA A DOI
  VECTORI
10 FOR I=1 TO 3
20      READ A(I)
30 NEXT I
40 DATA 6, 2, 1
50 FOR I=1 TO 3
60      READ B(I)
70 NEXT I
80 DATA 1, 2, 5
90 REM --------
100 FOR I=1 TO 3
110      LET C(I)=A(I)+B(I)
120 NEXT I
130 FOR I=1 TO 3
140      PRINT C(I);
150 NEXT I
160 END
```

b) Se consideră vectorii:

a=[3 8 2]

şi

b=[1 2 3].

Să se scrie un program BASIC care calculează produsul scalar

a·b=(3)(1)+(8)(2)+(2)(3)=25

```
5 REM * PRODUSUL SCALAR
  A DOI VECTORI *
10 FOR I=1 TO 3
20      READ A(I), B(I)
30 NEXT I
40 DATA 3, 1, 8, 2, 2, 3
50 REM - - - - - - - -
60 P=0
70 FOR I=1 TO 3
80      LET P=P+A(I) * B(I)
90 NEXT I
100 PRINT "PRODUSUL SCALAR=" ; P
110 END
```

IF–THEN

În limbajul BASIC–COMMODORE instrucţiunea **IF–THEN** prezintă următoarele **particularităţi:**

Format general
⟨nr. linie⟩ **IF** ⟨relaţie⟩ **THEN** { ⟨acţiune⟩[: ⟨acţiune 1⟩] . . . / ⟨nr. linie 1⟩ }

- ⟨relaţie⟩ este de forma unei relaţii între două expresii (numerice sau şiruri) sau a unei reuniuni de relaţii prin operatorii logici: **AND, NOT, OR**;
- ⟨acţiune⟩ poate fi o instrucţiune **GOTO** sau o altă instrucţiune. După **THEN** poate urma o secvenţă de instrucţiuni.

Aplicaţie. Precizaţi rezultatul execuţiei următoarelor programe:

a)

```
10 X=13
20 Y=40
30 IF X>10 AND Y>20
   THEN 50
40 PRINT "CONDIŢIILE NU
   SÎNT ÎNDEPLINITE" : END
50 PRINT "CONDIŢIILE SÎNT
   ÎNDEPLINITE"
RUN
CONDIŢIILE SÎNT ÎNDEPLINITE
```

b)

```
10 X=80
20 Y=15
30 IF Y=15 PRINT Y : PRINT X
RUN
15
80
```

În program instrucţiunile **IF–THEN** şi **GOTO** s-au utilizat pentru simularea structurii de iteraţie de tip **DO WHILE.**

Aplicaţii

a) Să se scrie un program care calculează e^x.

Indicaţie. Se va utiliza formula

$$e^x = \sum_{n=0}^{10} \cdot \frac{x^n}{n!}$$

```
10 REM APLICAŢIE
20      PRINT "X=", "EXP (X)=", "S="
30      A(1)=1
40      FOR I=1 TO 9
50           A(I+1)=A(I)/(I+1)
60      NEXT I
70      FOR K=1 TO 11
80           X=(K-1) * .1
90           S=A(10)
100          FOR I=1 TO 9
110               L=10-I
120               S=S * X+A(L)
```

```
130          NEXT I
140      S=S * X+1
150      P=EXP (X)
160      PRINT X, P, S
170      NEXT K
180 END
RUN
X=            EXP (X)=            S=
 0             1                  1
 .1            1.105171           1.105171
 .2            1.221403           1.221403
 .3            1.349859           1.349859
```

Remarcă. Funcţia aritmetică **EXP**(X) calculează e^x. Formatul general este

Format general
EXP (⟨expresie numerică⟩)

b) Introduceţi şi executaţi următoarele programe:

b.1)
```
10 PRINT CHR$ (147)          70 GOTO 10
20 A=0                       90 PRINT "⟨=⟩⟨=⟩⟨=⟩"
30 A=A+1                     100 B=0
40 IF A<100 THEN 30          110 GO TO 20
50 IF B=1    THEN 90         120 END
60 B=1
```

b.2)
```
10 PRINT CHR $ (147)         80 C=1
20 C=0                       90 GOTO 30
30 FOR I=1 TO 200            100 PRINT CHR$ (147)
40 NEXT I                    110 PRINT
50 IF C=1 THEN 100           120 PRINT "⟨=⟩"
60 PRINT CHR $ (147)         130 GO TO 20
70 PRINT "⟨=⟩"
```

□ Particularităţi ale programării în limbajul BASIC-80

vol. 2, pag. 214

Variabile indexate (numerice). Reguli

În limbajul BASIC-80 variabilele indexate (numerice) prezintă următoarele **caracteristici:**

- numele unei variabile indexate este orice nume de variabilă simplă acceptat de BASIC-80 urmat de una sau mai multe expresii întregi cuprinse între paranteze;
- este posibilă definirea prin acelaşi nume a unei variabile simple şi a unei variabile indexate;
- variabilele indexate pot fi de două tipuri: întreg, real. Tipul întreg se indică prin caracterul "%" plasat după numele simbolic al variabilei;
- valoarea minimă a unui indice este zero;
- pentru variabilele indexate se rezervă spaţiu, după cum urmează: elemente întregi – 2 octeţi/element; elemente simplă precizie – 4 octeţi/element; elemente dublă precizie – 8 octeţi/element.

Programul utilizează variabila indexată numerică cu un indice R(I).

DIM

În limbajul BASIC-80 instrucţiunea **DIM** prezintă următoarele **particularităţi:**

- dacă dimensiunea unei variabile indexate nu este specificată, atunci este stabilită implicit la 10;
- dacă dimensiunea maximă este depăşită se afişează un mesaj de eroare;
- instrucţiunea stabileşte pentru toate variabilele unui tablou, valoarea iniţială zero;
- variabilele tablou ale unui program pot fi eliminate cu instrucţiunea **ERASE** (v. BASIC-AMSTRAD).

IF–THEN–ELSE şi IF–GOTO–ELSE

În limbajul BASIC-80, instrucţiunea **IF–THEN–ELSE** (v. formatul general al instrucţiunii din limbajul BASIC-PRAE) prezintă următoarele **particularităţi:**

- ⟨relaţie⟩ este de forma unei relaţii între două expresii (numerice sau şiruri) sau a unei reuniuni de relaţii prin operatorii logici: **AND, OR, NOT, XOR** (sau exclusiv), **IMP** – implicaţie (X **IMP** Y=1 dacă X<=Y), **EQV** – echivalenţă (X **EQV** Y=1 dacă X=Y);
- Operatorii relaţionali admişi sînt: "="; "<"; ">"; "<=" sau "=<"; ">=" sau "=>"; "<>" sau "><";
- Este posibil ca o instrucţiune **IF** să includă o altă instrucţiune **IF**;
- Cînd se foloseşte **IF** pentru a testa mărimea rezultatului unei operaţii cu numere în virgulă mobilă, se va ţine seama de faptul că valoarea nu poate să fie exactă.

În următorul exemplu:

```
10 IF ABS (M–1.0) < 1.ØE–6 THEN 500
```

valoarea lui M este 1.0 cu o aproximaţie de 1E–6 (10^{-6}).

Aplicaţie. Se consideră următorul program BASIC:

```
10 GREŞIT=0
20 FOR I=1 TO 10
30    READ N1, N2
40    PRINT N1; "X"; N2; "=";
50    INPUT RĂSPUNS
60    IF RĂSPUNS <>N1 * N2 THEN GREŞIT=GREŞIT+1
70 NEXT
80 PRINT "AŢI DAT"; GREŞIT; "RĂSPUNSURI GREŞITE"
90 DATA 5, 5, 4, 4, 6, 6, 7, 7, 8, 8
100 DATA 4, 6, 4, 7, 5, 8, 5, 9, 6, 9
110 END
```

a) Introduceţi şi executaţi programul.

b) Adăugaţi instrucţiunile

```
65 IF RĂSPUNS <>N1 * N2 THEN PRINT "RĂSPUNS GREŞIT"
66 IF RĂSPUNS <>N1 * N2 THEN PRINT "RĂSPUNSUL ESTE"; N1 * N2
```

şi executaţi programul.

c) Adăugați instrucțiunile:

```
60 IF RĂSPUNS < > N1 * N2 THEN GREŞIT=GREŞIT+1 : PRINT
   "RĂSPUNS GREŞIT" : PRINT "RĂSPUNSUL ESTE"; N1 * N2
65
66
```

și executați programul.

d) Adăugați instrucțiunea

```
60 IF RĂSPUNS < > N1 * N2 THEN GREŞIT=GREŞIT+1 :
   PRINT "RĂSPUNS GREŞIT" : PRINT "RĂSPUNSUL ESTE";
   N1 * N2 ELSE PRINT "FELICITĂRI"
```

și executați programul.

TEST

Precizați rezultatele execuției următoarelor programe:

a)

```
10 X=8
20 Y=2
30 IF X>Y THEN PRINT "X>Y" ELSE IF
   Y>X THEN PRINT "Y>X" ELSE PRINT "X=Y"
```

R. X>Y

b)

```
10 A=2
20 B=5
30 C=7
10 IF A=B THEN IF B=C THEN PRINT "A=C" ELSE PRINT "A< >C"
```

R. A< >C

Observație. Limbajul BASIC-80 admite pentru instrucțiunea **IF** și formatul:

Format general

⟨nr. linie⟩ **IF** ⟨relație⟩ **GOTO** ⟨nr. linie 1/instrucțiune 1⟩ [**ELSE** ⟨nr. linie 2/instrucțiune 2⟩]

WHILE–WEND

WHILE–WEND creează un ciclu compus din mai multe instrucțiuni BASIC-80 atîta timp cît o condiție dată este adevărată. Formatul general și remarcile privind instrucțiunea **WHILE-WEND** sînt identice cu cele de la limbajul BASIC-AMSTRAD cu singura deosebire că ⟨secvență instrucțiuni⟩ reprezintă orice tip de instrucțiune BASIC-80.

În cadrul programului instrucțiunile **WHILE** și **WEND** din liniile 170 și 220 formează o buclă pentru validarea razelor rezervoarelor.

Înapoi la program

V-ați pus vreodată întrebarea ce s-ar întîmpla dacă pentru „Număr rezervoare" am introduce, evident din greșeală, un număr zecimal, 5.60 de exemplu! Vă rugăm să încercați chiar în acest moment. Ce-ați constatat?

Deși calculatorul nu ne semnalează nici o greșeală (!), totuși știm bine că există. Pentru a evita astfel de situații vă recomandăm să utilizați

o variabilă de tip întreg (N%) sau să introduceți în cadrul programului o secvență de instrucțiuni care să vă asigure că pentru „număr rezervor" s-a citit un număr întreg.

```
80 PRINT "Număr rezervoare=";
90 INPUT N
95 WHILE N-INT (N)< >0
96        INPUT "Număr rezervoare!"; N
98 WEND
```

Jocuri pe calculator

a) Introduceți și executați următoarele programe:

a.1)
```
10 RĂSPUNS=0
20 A=11
30 B=12
40 WHILE RĂSPUNS< >A*B
50    PRINT A; "X"; B; "=";
60    INPUT RĂSPUNS
70 WEND
80 END
```

a.2)
```
10 RĂSPUNS=91
20 WHILE RĂSPUNS>80
30   INPUT "INTRODUCEŢI UN
     NUMĂR<80"; RĂSPUNS
40 WEND
50 END
```

a.3)
```
10 SCOR=0
20 RĂSPUNS=0
30 A=15
40 WHILE RĂSPUNS< >A*A
     AND SCOR<4
50    PRINT A; "X"; A; "=";
60    INPUT RĂSPUNS
70    SCOR=SCOR+1
80 WEND
90 END
```

a.4)
```
10 INPUT "INTRODUCEŢI UN NU-
     MĂR DE LA 3 la 8"; N
20 WHILE N<3 OR N>8
30    INPUT "INTRODUCEŢI UN
      ALT NUMĂR"; N
40 WEND
50 PRINT "FELICITĂRI!"
60 END
```

a.5)
```
5 PRINT "PENTRU OPRIRE
     TASTAŢI A=999"
10 INPUT "A, B"; A, B
20 WHILE A< >999
30    IF A>B THEN PRINT "A>":
PRINT "B": GOTO 60
40    IF A=B THEN PRINT "A=":
      PRINT "B": GOTO 60
50    PRINT "A<": PRINT "B"
60    PRINT
70    INPUT "A, B"; A, B
80 WEND
90 END
RUN
A; B ? 4, 6
A<
B
A, B? 80, 80
A=
B
A, B? 98, 2
A>
B
A, B? 999
? Redo from start
A, B? 999,0
OK
```

a.6)
```
20 Y=0
30 Z=Y↑2
40 IF Z>48 THEN END
50 PRINT TAB (Z); "<=>"
60 Y=Y+1
70 GO TO 30
```

Observație. Același program (a.6) poate fi scris și sub forma:

```
20 Y=0
30 Z=Y↑2
40 WHILE Z<48
50    PRINT TAB (Z); "<=>"
60    Y=Y+1
70    Z=Y↑2
80 WEND
90 END
```

b) Scrieți un program BASIC pentru înmulțirea fracțiilor:

$$\frac{1}{2} \cdot \frac{4}{3}; \quad \frac{2}{9} \cdot \frac{6}{8}$$

```
10 READ N1; D1
20 DATA 1, 2
30 WHILE N1< >999
40       READ N2, D2
50       DATA 4, 3
60       N=N1*N2
```

```
70   N3=N                    160     R=N-I*D
80   D=D1*D2                 170   WEND
90   D3=D                    180   PRINT "PRODUSUL
100  I=INT (N/D)                   ESTE"; N3/D; "/"; D3/D
110  R=N-I*D                 190   READ N1, D1
120  WHILE R< >0             200 WEND
130    N=D                   210   DATA 2, 9, 6, 8
140    D=R                   220   DATA 999, 0
150    I=INT(N/D)            230   END
```

TEST

Precizaţi rezultatul execuţiei următorului program:

```
10 READ A, B                        60 WEND
15 DATA 3, 4                        70 DATA 35, 7, 99, 34, 80, 25
20 WHILE A< >9999                   80 DATA 838, 21, 97, 13, 80, 40
30   R=A-INT(A/B)*B                 90 DATA 9999, 0
40   PRINT A; B; R; INT (A/B)      100 END
50 READ A, B
```

GOSUB şi RETURN

Să nu uităm de modulele comune de prelucrare: RÎND şi CIT-R (v. figurile 5.3 şi 5.4) care aşteaptă momentul programării lor în BASIC. Cum vom proceda? Pentru început, trebuie să ne preocupe locul de plasare a acestora.

Practica demonstrează necesitatea plasării lor la sfîrşitul programului (v. instrucţiunea **STOP**), între **STOP** şi **END**, avînd grijă ca ultima instrucţiune a fiecărui modul (putem să-l numim subprogram sau subrutină) să fie instrucţiunea **RETURN**. De notat că o subrutină BASIC nu are nume. Cum poate fi atunci apelată? Subrutinele pot fi apelate din orice punct al programului, chiar şi din cadrul unei alte subrutine (pot exista situaţii cînd o subrutină se poate reapela) cu ajutorul instrucţiunii **GOSUB** al cărei format general este:

Formatul general

```
                 GOSUB ⟨număr linie⟩
                   .
⟨număr linie⟩      .
                   .
                   .
                 RETURN
```

în care ⟨număr linie⟩ reprezintă numărul de linie al primei instrucţiuni din cadrul subrutinei.

Mecanismul (v. figura 5.11) este următorul. Instrucţiunea **GOSUB** lansează în execuţie subrutina care începe la numărul de linie indicat în formatul general prin ⟨număr linie⟩ şi reţine instrucţiunea care urmează în secvenţă. Instrucţiunea **RETURN** marchează sfîrşitul execuţiei subrutinei şi redă controlul programului principal, urmînd a se executa instrucţiunea imediat următoare după **GOSUB**.

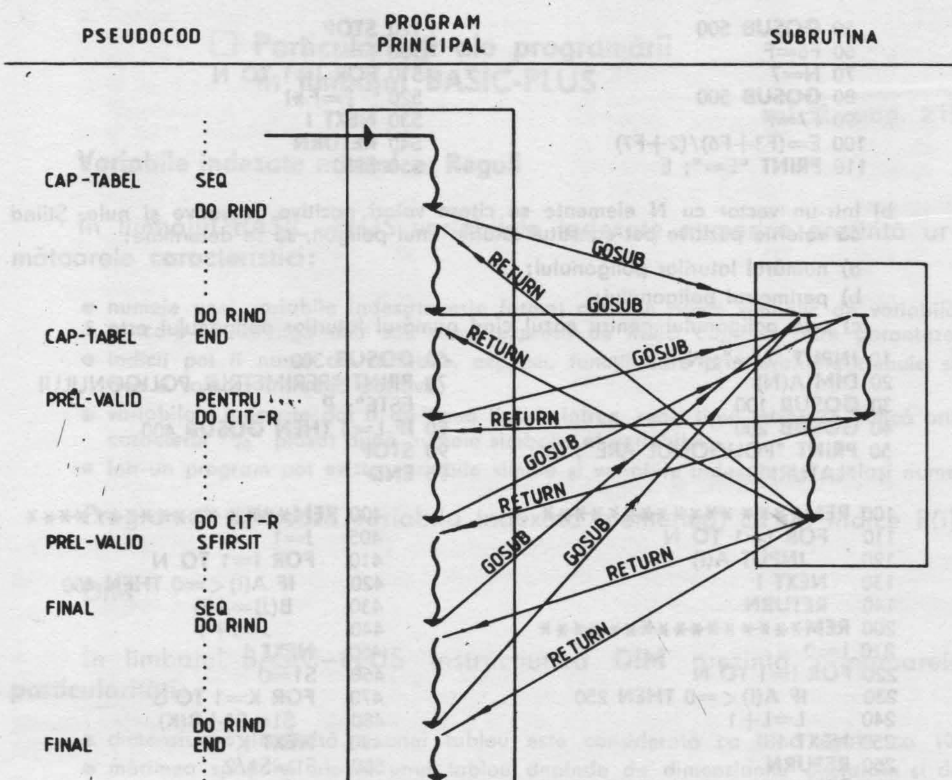


Fig. 5.11. Mecanismul de funcţionare a instrucţiunilor **GOSUB** şi **RETURN.**

Remarci

- Subrutinele sînt folosite pentru a realiza operaţii care se repetă frecvent.
- Avantajele lucrului cu subrutinele sînt: economisirea de timp la scrierea programului, reducerea consumului de memorie, creşterea gradului de lizibilitate a programului etc.
- O subrutină nu are argumente, schimbul de informaţii cu programul apelant realizîndu-se prin variabilele curente.
- Subrutina trebuie să conţină cel puţin o instrucţiune **RETURN.**
- Subrutina este activată în momentul în care în programul principal s-a citit un **GOSUB.**
- Instrucţiunea **RETURN** dezactivează subrutina.
- Subrutinele pot fi imbricate.
- Revenirea dintr-o subrutină apelată într-o subrutină apelantă se realizează cu respectarea regulii revenirii la punctul imediat următor punctului de ieşire.
- Profunzimea imbricării subrutinelor depinde numai de spaţiul de memorie disponibil.

Aplicaţii

a) Să se calculeze expresia:

$$E=\frac{3!+6!}{2+7!}$$

```
10 N=3
20 GOSUB 500
30 F3=F
40 N=6
```

```
50 GOSUB 500
60 F6=F
70 N=7
80 GOSUB 500
90 F7=F
100 E=(F3+F6)/(2+F7)
110 PRINT "E="; E
120 STOP
500 F=1
510 FOR I=1 TO N
520   F=F*I
530 NEXT I
540 RETURN
550 END
```

b) Într-un vector cu N elemente se citesc valori pozitive, negative şi nule. Ştiind că valorile pozitive pot constitui laturile unui poligon, să se determine:

a) numărul laturilor poligonului;

b) perimetrul poligonului;

c) aria poligonului pentru cazul cînd numărul laturilor poligonului este 3.

```
10 INPUT "N="; N
20 DIM A(N)
30 GOSUB 100
40 GOSUB 200
50 PRINT "POLIGONUL ARE"; L;
   "LATURI"
60 GOSUB 300
70 PRINT "PERIMETRUL POLIGONULUI
   ESTE"; P
80 IF L=3 THEN GOSUB 400
90 STOP
95 END

100 REM****************
110   FOR I=1 TO N
120     INPUT A(I)
130   NEXT I
140   RETURN
200 REM****************
210 L=0
220 FOR I=1 TO N
230     IF A(I)<=0 THEN 250
240     L=L+1
250 NEXT I
260 RETURN
300 REM****************
305 P=0
310 FOR I=1 TO N
320     IF A(I)<=0 THEN 340
330     P=P+A(I)
340 NEXT I
400 REM****************
405   J=1
410   FOR I=1 TO N
420     IF A(I)<=0 THEN 450
430     B(J)=A(I)
440       J=J+1
450   NEXT I
460   S1=0
470   FOR K=1 TO 3
480     S1=S1+B(K)
490   NEXT K
500   S1=S1/2
510     S1=1
520   FOR K=1 TO 3
530     S1=S1 * (S1-B(K))
540   NEXT K
550   S1=S1 ↑ 0.5
560   PRINT "ARIA="; S1
570   RETURN
```

c) Scrieţi un program pentru adunarea a două fracţii.

```
5   REM ADUNAREA A DOUĂ
                 FRACŢII
10  REM A/B+C/D
20  INPUT "A, B, C, D"; A, B, C, D
25  X=A
30  Y=B
40  GOSUB 400
50  D2=E
60  A1=A/D2
70  B1=B/D2
80  X=C
90  Y=D
100 GOSUB 400
110 D2=E
120 C1=C/D2
130 D1=D/D2
140 X=B1
150 Y=D1
160 GOSUB 400
170 D2=E
180 N3=(B1*D1)/D2
190 N4=(N3/B1)*A1+(N3/D1)*C1
200 PRINT N4
210 PRINT N3
220 X=N4
230 Y=N3
240 GOSUB 400
250 D2=E
260 F1=N4/D2
270 F2=N3/D2
280 PRINT A, B, C, D
290 PRINT F1, F2
300 STOP
320 END
400 E=X
410 F=Y
420 IF F=0 THEN 470
430 R=E-INT(E/F)*F
440 E=F
450 F=R
460 GOTO 420
470 RETURN
```

□ Particularităţi ale programării în limbajul BASIC-PLUS

vol. 2, pag. 215

Variabile indexate numerice. Reguli

În limbajul BASIC–PLUS variabilele indexate numerice prezintă următoarele **caracteristici**:

- numele unei variabile indexate este format dintr-un nume simbolic de variabilă, la care se adaugă una sau două expresii de indici cuprinse între paranteze;
- indicii pot fi nume de variabile, expresii, funcţii care prin evaluare trebuie să dea o valoare întreagă pozitivă;
- variabilele indexate pot fi de două tipuri: întreg, real; tipul întreg se indică prin caracterul "%" plasat după numele simbolic al variabilei;
- într-un program pot exista variabile simple şi variabile indexate cu acelaşi nume.

Programul utilizează variabila indexată (numerică) cu un indice R(I).

DIM

În limbajul BASIC–PLUS instrucţiunea **DIM** prezintă următoarele **particularităţi**:

- dimensiunea implicită a unui tablou este considerată ca fiind egală cu 10;
- mărimea spaţiului alocat unui tablou depinde de dimensiunile declarate şi de tipul său (întreg, real, şir) şi se calculează cu formula:

$$S=\langle dim\ 1\rangle * \langle dim\ 2\rangle * K \text{ octeţi}$$

în care:
⟨dim 1⟩ şi ⟨dim 2⟩ reprezintă dimensiunile tabloului şi pot fi expresii numerice (întregi sau reale);

$$K=\begin{cases} 2, \text{ pentru tablouri numerice întregi,} \\ 4, \text{ pentru tablouri numerice reale sau şir.} \end{cases}$$

Observaţie. În calculul anterior nu intră spaţiul ocupat de elementele tabloului şir de caractere.

- numărul tablourilor ce pot fi utilizate într-un program este funcţie numai de spaţiul (maxim) necesar şi de spaţiul de memorie aflat la dispoziţia utilizatorului;
- tabloul nu este alocat în momentul declarării lui ci atunci cînd se tratează efectiv un element al său; după alocare el poate fi redimensionat avînd în vedere dimensiunea spaţiului fizic rezervat.

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 C=3·14159/180
20 FOR I=0 TO 80 STEP 10
30    PRINT SIN (I*C); COS (I*C); TAN (I*C)
40 NEXT I
50 END
```

Observaţie. **SIN, COS, TAN** calculează sinusul, cosinusul şi tangenta expresiei I*C. Argumentele funcţiilor trigonometrice au fost transformate în radiani (v. linia 10).

TEST

Precizați rezultatele execuției următoarelor programe BASIC:

a)
```
10 DIM A(50)
20 FOR I=1 TO 50
30   A(I)=I
40 NEXT I
50 NEXT I
60 FOR I=1 TO 50
70   PRINT I/2,
80 NEXT I
```

b)
```
10 DIM A(5)
20 DATA 8, -7, 5, -6, 4
30 FOR I=1 TO 5
40   READ A(I)
50 NEXT I
```

c)
```
10 FOR I=1 TO 10
20   FOR K=1 TO 20
30     PRINT I+K,
40   NEXT K
```

d)
```
10 FOR I=1 TO 10
20   FOR K=1 TO 20
30     PRINT 1+K,
40   NEXT K
50 NEXT I
60 FOR I=1 TO 50
70   FOR I=1 TO 10
80     PRINT I/2
90   NEXT I
100 NEXT I
```

IF–THEN–ELSE și IF–GOTO–ELSE

În limbajul BASIC–PLUS instrucțiunile **IF–THEN–ELSE** și **IF–GOTO–ELSE** (v. formatul general al instrucțiunilor din limbajul BASIC-80) prezintă următoarele **particularități:**

- ⟨relație⟩ este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații prin operatorii logici: **AND, OR, NOT, XOR, IMP, EQV;**
- operatorii relaționali sînt: "="; "<"; "<="; ">"; ">="; "#" (diferit);
- ⟨instrucțiune 1⟩/⟨instrucțiune 2⟩ nu poate fi: **DIM, DEF, DATA, REM, FNEND, END, COM, FOR, NEXT.**

Aplicație. Introduceți și executați următoarele programe (unele incomplete!):

a)
```
10 REM PROGRAMUL EFECTUEAZĂ
   PRODUSUL
20 REM MAI MULTOR PERECHI DE
   NUMERE (N)
30 PRINT "INTRODUCEȚI NUMĂRUL
   DE PERECHI DE NUMERE";
40 INPUT N
50 FOR I=1 TO N
60   INPUT "A, B"; A, B
70   IF A/2=INT (A/2) THEN GOTO ?
80   LET P=P+B
90   LET A= INT (A/2)
100  LET B=2*B
110  IF A>=1 THEN GOTO ?
120  PRINT "PRODUSUL ESTE:"; P
130 NEXT I
140 END
RUN
INTRODUCEȚI NUMĂRUL DE PERECHI
DE NUMERE ? 1
A, B ? 17, 12
PRODUSUL ESTE: 204
```

b)
```
5 REM 1+2+3+4+5+6+7+8+9+
  +10
10 READ N
20 DIM X(N)
30 FOR I=1 TO N
40   READ X(I)
50 NEXT I
60 DATA 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
70 S=0
80 FOR I=1 TO N
90   S=S+X(I)
100 NEXT I
110 PRINT "SUMA NUMERELOR DE LA
    1 LA 10 ESTE"; S
120 END
```

c)
```
5 REM 1×2×3×4×5×6×7×8×
  ×9×10
10 READ N
20 DIM X(N)
30 FOR I=1 TO N
40   READ X(I)
50 NEXT I
60 DATA 10
70 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
80 P=1
90 FOR I=1 TO N
100  IF X(I)=0 THEN PRINT "P=0" :
     END
110  P=P*X(I)
120 NEXT I
130 PRINT "1×2×3×4×5×6×7×8×
    ×9×10="; P
```

d)
```
10 INPUT "PRIMUL NUMĂR"; A
20 INPUT "AL DOILEA NUMĂR"; B
30 PRINT
40 IF NOT (A>B) THEN PRINT
   "A=SAU<B": GOTO 80
50 IF NOT (A=B) THEN PRINT
   "A(SAU)B": GO TO 80
60 PRINT "A=SAU>B"
70 PRINT A, B
80 PRINT : PRINT
90 PRINT "TASTAŢI 0 DACĂ
   NUMAI CONTINUAŢI"
100 PRINT "TASTAŢI ORICE NUMĂR
   PENTRU CONTINUARE"
110 INPUT N
120 IF NOT (N=0) THEN 10
130 PRINT
140 END
```

e)
```
10 A%=120
20 PRINT ">>>";
30 A%=A%-1
40 IF A%=20 THEN PRINT : PRINT
   TAB (10); "A%=20" : PRINT
50 IF A%>-1 GO TO 20
60 PRINT
```

f)
```
10 AREAL = .999999999
20 LINIE% = 1
30 PRINT "NR. LINIE", "REAL"
40 PRINT
50 PRINT LINIE%, AREAL
60 AREAL = AREAL * 10
70 LINIE% = LINIE% + 1
80 GO TO 50
90 END
```

g)
```
10 DIM F(20)
20 PRINT
30 F(1)=1
40 F(2)=1
50 FOR I=3 TO 20
60    F(I)=F(I-1)+F(I-2)
70 NEXT I
80 FOR I=1 TO 20
90       PRINT F(I),
100 NEXT I
110 PRINT
220 END
```

Înapoi la program

În cadrul programului controlul logic formal asupra datelor de intrare este asigurat prin intermediul instrucţiunilor **IF–THEN** (v. linia 170) şi **GO TO** (v. linia 190). Mesajul "EROARE DE INTRODUCERE DATE ***** RAZA NEGATIVĂ **** RELUARE" oferă posibilitatea de a identifica valoarea negativă a razei şi de a o corecta prin intermediul instrucţiunii **INPUT** (v. linia 140).

```
100 REM CITIREA DINAMICĂ ŞI VALIDARE DATE
110 DIM R(N)
120 FOR I=1 TO N
130     PRINT "REZERVORUL NUMĂRUL"; I
140     INPUT "RAZA REZERVORULUI"; R(I)
150     PRINT "AŢI SPECIFICAT"; R(I)
160     PRINT
170     IF R(I) > 0 THEN 200
180     PRINT "EROARE DE INTRODUCERE DATE * RAZA NEGATIVĂ * RELUARE"
190     GO TO 130
200 NEXT I
```

Cu valorile corecte ale razelor se calculează şi însumează cantitatea de benzină din rezervoare. Conversaţia purtată de noi cu un minicalculator CORAL prin intermediul programului BASIC–PLUS este:

```
:RUN
 NUMĂR REZERVOARE =?                      6

 AŢI SPECIFICAT                           6 REZERVOARE
REZERVORUL NUMĂRUL                          1
RAZA REZERVORULUI                          ?2
AŢI SPECIFICAT                              2

REZERVORUL NUMĂRUL                          2
RAZA REZERVORULUI                          ?-2.3
AŢI SPECIFICAT                             -2.3
```

```
EROARE DE INTRODUCERE DATE * * * * * RAZA NEGATIVĂ * * * * RELUARE
REZERVORUL NUMĂRUL              2
RAZA REZERVORULUI               ?2.3
AȚI SPECIFICAT                  2.3
REZERVORUL NUMĂRUL              3
RAZA REZERVORULUI               ?4
AȚI SPECIFICAT                  4
REZERVORUL NUMĂRUL              4
RAZA REZERVORULUI               ?6.2
AȚI SPECIFICAT                  6.2
REZERVORUL NUMĂRUL              5
RAZA REZERVORULUI               ?-3
AȚI SPECIFICAT                  -3
EROARE DE INTRODUCERE DATE * * * * * RAZA NEGATIVĂ * * * * RELUARE
REZERVORUL NUMĂRUL              5
RAZA REZERVORULUI               ?3
AȚI SPECIFICAT                  3
REZERVORUL NUMĂRUL              6
RAZA REZERVORULUI               ?10.0
AȚI SPECIFICAT                  10
CALCULUL MASEI TOTALE DE BENZINĂ
DIN 6 REZERVOARE CILINDRICE ECHILATERALE
RAZA          MASA
2             35.1858
2.3           53.5133
4             281.487
6.2           1048.22
3             118.752
10            4398.23
MASA TOTALĂ = 5 935.39 TONE
...........................................
STOP AT LINE 440
```

Aplicație. Să se scrie un program BASIC care calculează cos (x) cu relația:

$$\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots+(-1)^{n+1}\cdot\frac{x^{2n-2}}{(2n-2)!}$$

```
10 PRINT "X=";
20 INPUT X
30 S1=S=N=F=1
40 PRINT N ; S
50 N=N+1
60 S1=S
70
80 F=F * (2 * N-2) * (2 * N-3)
90 S=S+((-1) ↑ (N+1) * X ↑ (2 * N-2)/F)
100 IF ABS (S-S1)/(ABS (S)+ ABS (S1)) > 1E-6 THEN 40
110 PRINT "COSINUS CALCULAT CU COS (X)" ; COS (X)
120 END
```

☐ Particularități ale programării în limbajul ABASIC

vol. 2, pag. 216

Variabile indexate numerice. Reguli

În limbajul ABASIC variabilele indexate prezintă următoarele **caracteristici:**

- numele unei variabile indexate este alcătuit dintr-un nume admis de variabila simplă urmat de una, două sau mai multe expresii de indici cuprinse între paranteze;
- indicii pot fi nume de variabile, expresii, funcţii care prin evaluare trebuie să dea o valoare întreagă pozitivă;
- variabilele indexate pot fi de două tipuri: intreg, real; tipul întreg se indică prin caracterul " % " plasat după numele simbolic al variabilei;
- într-un program pot exista variabile simple şi variabile indexate cu acelaşi nume;
- valoarea minimă a unui indice este zero sau unu (v. opţiunea **BASE**);

În cadrul programului variabila indexată numerică (cu un indice) este R(I).

DIM

În limbajul ABASIC instrucţiunea **DIM** prezintă următoarele **particularităţi**:

- numărul de dimensiuni ale tablourilor declarate este limitat numai de lungimea liniei ABASIC (80 de caractere);
- pentru a preciza rangul primului element al unui tablou (0 sau 1) utilizatorul dispune de instrucţiunea **OPTION BASE** al cărei format este:

Format general
OPTION BASE ⟨expresie⟩

în care ⟨expresie⟩ poate lua valorile 0 sau 1.

IF

În limbajul ABASIC instrucţiunea **IF** prezintă trei formate generale.

Formatul – 1: IF–THEN–ELSE

Instrucţiunea **IF–THEN–ELSE** prezintă acelaşi format ca şi instrucţiunea omoloagă din limbajul BASIC-80, cu următoarele **particularităţi:**

- operatorii relaţionali admişi sînt: "="; "<"; ">"; "<="; ">="; "<>" şi "==" (aproximativ egal);
- operatorii logici admişi sînt: **NOT, AND, OR, XOR, EQV, IMP.**

Observaţie. Pentru variabilele numerice, operatorul "==" are semnificaţia unei egalităţi aproximative, cu o diferenţă mai mică de 10^{-8}.

- instrucţiunea (**IF–THEN–ELSE**) trebuie să fie conţinută pe o singură linie program;
- dacă o instrucţiune **IF** este urmată pe aceeaşi linie de alte instrucţiuni **IF** atunci fiecare **ELSE** corespunde **IF**-ului cel mai apropiat;
- ⟨instrucţiune 1⟩/⟨instrucţiune 2⟩ poate fi orice tip de instrucţiune care admite scrierea pe o singură linie program.

Aplicaţie. Scrieţi un program BASIC care generează şi tipăreşte toate punctele pitagorice cu valori ale ipotenuzei mai mici ca 101.

```
10 FOR I=3 TO 98
20   FOR J=I+1 TO 99
30     K1=SQR (I * I+J * J)
40     IF K1 <> INT (K1) THEN 70
50     IF K1 > 100 THEN 80
60     PRINT I; J; K1
70   NEXT J
80 NEXT I
90 END
RUN
```

Formatul – 2: IF–GOTO

Instrucţiunea IF–GO TO are formatul:

Format general
⟨nr. linie⟩ IF ⟨relaţie⟩ GOTO ⟨nr. linie⟩

în care ⟨relaţie⟩ şi ⟨nr. linie⟩ au semnificaţiile cunoscute.

Observaţie. Instrucţiunea trebuie să fie conţinută pe o singură linie program.

Formatul – 3: IF–THEN–ELSE–ENDIF

Instrucţiunea are formatul:

Format general
⟨nr. linie⟩ IF ⟨relaţie⟩ [THEN] ⟨bloc – instrucţiuni 1⟩ [ELSE ⟨bloc – instrucţiuni 2⟩] ENDIF

în care ⟨bloc instrucţiuni⟩ reprezintă o secvenţă de instrucţiuni ABASIC, scrise pe una sau mai multe linii program.

Remarcă. Instrucţiunea poate ocupa mai multe linii terminal avînd în vedere restricţiile de mai jos:
- cuvîntul cheie **ELSE** (dacă este folosit) trebuie să fie singur pe linia program;
- cuvîntul cheie **ENDIF** trebuie să fie singur pe linia program;
- **IF** trebuie să fie ultima instrucţiune în linie.

Aplicaţie. Să se verifice dacă punctele A(x1, y1), B(x2, y2) şi C(x3, y3) sint coliniare. Programul va afişa unul din mesajele: "PUNCTE COLINIARE"; "PUNCTE NECOLINIARE"; "PUNCTE VERTICAL COLINIARE".
Specificaţiile de programare şi documentaţia de proiectare sint prezentate în modulele de analiză şi proiectare structurată, fig. 5.12.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieşire
X1,Y1: coordonatele punctului A	D1: diferenţa absciselor punctelor A şi B	M1: panta dreptei determinate de punctele A şi B
X2,Y2: coordonatele punctului B	D2: diferenţa ordonatelor punctelor B şi C	M2: panta dreptei determinate de punctele B şi C
X3,Y3: coordonatele punctului C		

a)

DATELE DE INTRARE

X1	Y1	X2	Y2	X3	Y3
6	5	0	7	–9	10
1	2	3	4	5	7
1	2	1	4	2	3
5	2	5	–3	5	50
999	0				

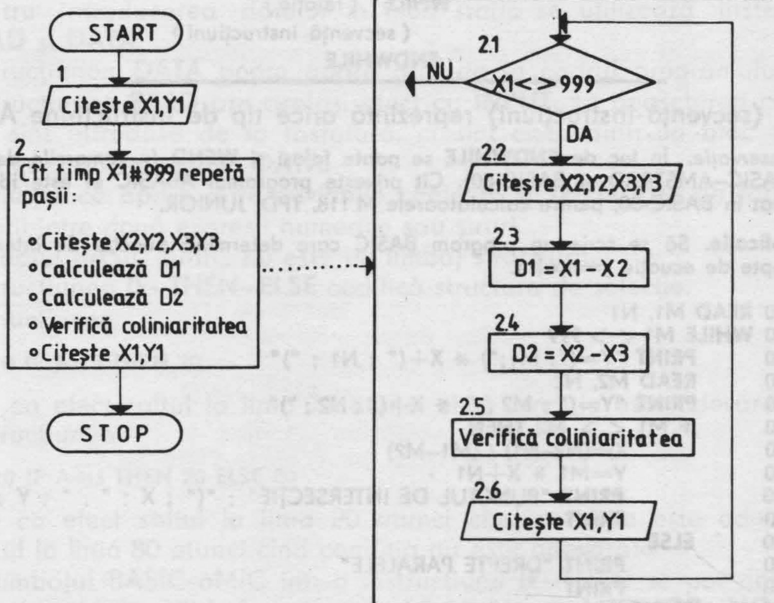
b)

Fig. 5.12. Modulele de analiză şi proiectare structurată: *a)* tabela de variabile; *b)* datele de intrare;

PSEUDOCODUL

```
APLIC   SEQ
        READ X1, Y1
WHILE   CÎT TIMP X1 < > 999
W       SEQ
           READ X2, Y2, X3, Y3
           PRINT X1, Y1, X2, Y2, X3, Y3
           D1=X1-X2
           D2=X2-X3
W       END
D       DACA D1 * D2 < > 0
        M1=(Y1-Y2) / D1
        M2=(Y2-Y3) / D2
D1      DACA M1 < > M2
           PRINT ""PUNCTE NECOLINAIRE"
D1      ÎN CAZ CONTRAR
           PRINT "PUNCTE NECOLINIARE"
D1      SFÎRŞIT
D       ÎN CAZ CONTRAR
D2         DACA D1 < > 0
              PRINT "PUNCTE NECOLINIARE"
D2         ÎN CAZ CONTRAR
D3         DACA D2 < > 0
                 PRINT "PUNCTE NECOLINIARE"
D3         ÎN CAZ CONTRAR
                 PRINT "PUNCTE VERTICAL COLINIARE"
D3         SFÎRŞIT
D2         SFÎRŞIT
D       SFÎRŞIT
        READ X1, Y1
WHILE   SFÎRŞIT
APLIC   END
```

c)



d)

c) pseudocodul; *d)* schema logică.

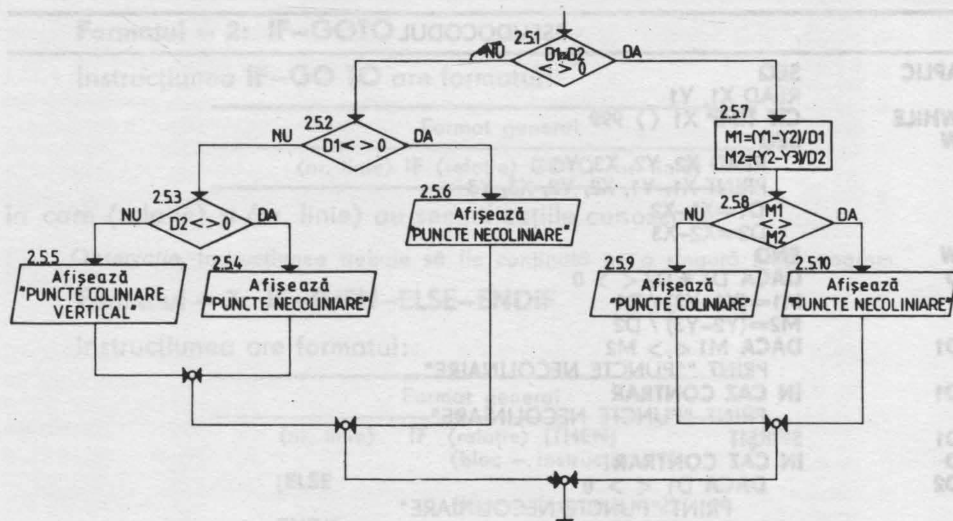


Fig. 5.12. d) schemă logică.

WHILE–ENDWHILE

WHILE–ENDWHILE creează un ciclu compus din mai multe instrucţiuni ABASIC atîta timp cît o condiţie dată este adevărată.

Formatul general al instrucţiunii este:

Format general
WHILE ⟨ relaţie ⟩
⟨ secvenţă instrucţiuni ⟩
ENDWHILE

în care ⟨secvenţă-instrucţiuni⟩ reprezintă orice tip de instrucţiune ABASIC.

Observaţie. În loc de **ENDWHILE** se poate folosi şi **WEND** (v. remarcile de la limbajele BASIC–AMSTRAD şi BASIC-80). Cît priveşte programul ABASIC el este identic cu cel realizat în BASIC-80, pentru calculatoarele M 118, TPD, JUNIOR.

Aplicaţie. Să se scrie un program BASIC care determină punctul de intersecţie a două drepte de ecuaţie y=mx+n.

```
10 READ M1, N1
20 WHILE M1 < > 999
30      PRINT "Y=("; M1;") * X+(" ; N1 ; ")"
40      READ M2, N2
50      PRINT "Y=(" ; M2 ; ") * X+(" ; N2 ; ")"
60      IF M1 < > M2 THEN
70           X=(N2-N1) / (M1-M2)
80           Y=M1 * X+N1
90           PRINT "PUNCTUL DE INTERSECŢIE" ; "(" ; X ; " , " ; Y ; ")"
100          PRINT
110     ELSE
120          PRINT "DREPTE PARALELE"
130          PRINT
140     ENDIF
```

```
150 READ M1, N1
160 WEND
170 DATA 3, -7, -4, 14, 1, 2, 3, 4,
180 DATA 5, 3, 5, 8, 2, -11, 5, -11
190 DATA 999,0
200 END
```

Observaţie. Variabilele de intrare M1, N1, M2, N2 reprezintă panta şi termenul liber pentru fiecare ecuaţie în parte. X, Y (variabilele de ieşire) reprezintă coordonatele punctului de intersecţie.

TEMA 5

☐ Răspundeţi prin DA sau NU la următoarele întrebări:

- O variabilă indexată se declară folosind instrucţiunea **DIM**.
- Într-o instrucţiune **DIM** se specifică între paranteze numărul de elemente din tablou.
- În BASIC-aMIC numărul de indici ai unei variabile indexate este 1 sau 2 (vector sau matrice).
- În BASIC-PRAE numărul de indici ai unei variabile indexate este 1 la 255.
- În aceeaşi instrucţiune **DIM** tablourile numerice şi cele şir pot fi declarate intercalat.
- Indicele (indexul) unei variabile indexate poate fi o constantă, o variabilă numerică simplă (de tip întreg) sau expresii aritmetice.
- Tablourile numerice pot avea una, două sau mai multe dimensiuni.
- Pentru introducerea datelor în mod static se utilizează instrucţiunile **READ** şi **DATA**.
- Instrucţiunea **DATA** poate apare oriunde în cadrul programului.
- Instrucţiunea **READ** are acelaşi efect cu **INPUT**, cu deosebirea că datele nu sînt introduse de la tastatură, ci sînt citite dintr-un bloc de date, definit cu instrucţiunea **DATA**.
- Condiţia ce apare între cuvintele cheie **IF** şi **THEN** este de forma unei relaţii între două expresii numerice sau şiruri.
- Limbajul BASIC-aMIC nu este un limbaj structurat.
- Instrucţiunea **IF–THEN–ELSE** codifică structura de selecţie.
- Instrucţiunea

  ```
  10 IF A<5 THEN 70
  ```

  are ca efect saltul la linia 70 atunci cînd condiţia este adevărată.
- Instrucţiunea

  ```
  10 IF A=3 THEN 20 ELSE 80
  ```

  are ca efect saltul la linia 20 atunci cînd condiţia este adevărată şi saltul la linia 80 atunci cînd condiţia nu este adevărată.
- În limbajul BASIC-aMIC într-o instrucţiune **IF–THEN** se pot defini condiţii compuse utilizînd operatorii logici binari **AND, OR, NOT.**

– Dacă două relaţii sînt reunite prin operatorul logic **AND** perechea relaţiilor este adevărată numai dacă ambele relaţii sînt adevărate.
– Dacă două relaţii sînt reunite prin operatorul logic **OR** perechea relaţiilor este adevărată dacă cel puţin una dintre relaţii este adevărată.
– În BASIC-PRAE pot fi declarate tablouri cu dimensiuni variabile.
– În BASIC HC-85, TIM S, SPECTRUM numele unei variabile indexate este alcătuit dintr-o singură literă.
– În BASIC HC-85, TIM S, SPECTRUM variabilele şir şi variabilele tablou-şir nu pot avea acelaşi nume.
– În BASIC-TIM S o variabilă numerică simplă şi o variabilă indexată pot avea acelaşi nume.
– În BASIC-SPECTRUM cu o instrucţiune **DIM** poate fi definită numai o singură variabilă indexată.
– În BASIC HC-85, TIM S, SPECTRUM operatorii logici **(AND, OR, NOT)** pot fi consideraţi şi folosiţi ca funcţii.
– În BASIC-AMSTRAD valoarea minimă a unui indice este zero.
– În BASIC-aMIC variabilele indexate pot fi de tip întreg şi real.
– În BASIC-PRAE, dacă dimensiunea **(DIM)** unei variabile indexate nu este specificată, ea se consideră implicit ca fiind 10.
– În limbajul BASIC-AMSTRAD există instrucţiunile **WHILE–WEND.**
– **WHILE–WEND** implementează structura de iteraţie **CIT–TIMP.**
– În BASIC–COMMODORE variabilele indexate pot fi de două tipuri: întreg şi real.
– În BASIC-80 variabilele tablou ale unui program pot fi eliminate cu instrucţiunea **ERASE.**
– În limbajul BASIC–PLUS se poate utiliza instrucţiunea **IF–GOTO–ELSE.**
– În limbajul ABASIC instrucţiunea **IF** prezintă trei formate generale.
– În limbajul BASIC-80 în loc de **WEND** se poate folosi şi **ENDWHILE.**

☐ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Un vector este ....
b) Structura **CÎT TIMP** se mai numeşte şi .... deoarece ....
Structura mai este cunoscută şi sub denumirea de „bucla de tip **DO** ....
c) Variabilele AGA (2), B (7) sînt variabile .... iar AGA (2) şi B (7) sînt variabile ....
d) Într-un program BASIC ...., pot exista variabile .... şi variabile indexate cu acelaşi nume.
e) Pentru introducerea dinamică a vectorilor de date se foloseşte instrucţiunea .... plasată într-o buclă **FOR–NEXT.**
f) Pentru introducerea statică a vectorilor de date se folosesc instrucţiunile ...., ... plasate într-o buclă **FOR–NEXT.**
g) Instrucţiunea .... are acelaşi efect cu **INPUT,** cu deosebirea că datele nu sînt introduse de la tastatură, ci sînt citite dintr-un .... definit cu instrucţiunea.................................
h) În BASIC structurat bucla de tip .... se codifică cu instrucţiunile **WHILE–WEND,** unde **WHILE** indică .... iar **WEND** ....

i) Pentru simularea iteraţiei **CIT TIMP** în limbajele BASIC ...., se aplică următoarea metodologie: ....
j) Instrucţiunea **IF–THEN** este o instrucţiune de salt .... iar instrucţiunea .... defineşte un salt necondiţionat.
k) Structura de selecţie poate fi cu .... sau cu .... alternative.
l) Structura **IF–THEN–ELSE** este specifică limbajului BASIC implementat pe calculatoarele ...., ... iar **IF–GOTO–ELSE** ....
m) Instrucţiunea **IF–THEN** ⟨nr. linie⟩ este specifică limbajului BASIC implementat pe calculatoarele .... iar **IF–THEN** ⟨nr. linie/instrucţiune⟩
n) Operatorii logici **AND, NOT, OR** sînt admişi de limbajul BASIC implementat pe calculatoarele ...., ..., ...
o) Operatorii logici **EQV** şi **IMP** sînt admişi de limbajul BASIC implementat pe calculatoarele ...., ...
p) Operatorul relaţional "= =" (aproximativ egal) este admis de limbajul BASIC implementat pe calculatoarele ...., ...
r) Instrucţiunea **IF–THEN–ELSE–ENDIF** este specifică limbajului BASIC implementat pe calculatoarele ...., ..., ...
s) În limbajul BASIC implementat pe calculatoarele ...., ... structura de iteraţie .... se codifică cu instrucţiunile **WHILE–WEND.**

☐ Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Să se efectueze împărţirea:

$$\frac{x^4+2x^3-13x^2-14x+24}{x-1}$$

b) Ca urmare a aplicării metodei de determinare a diametrelor medii aritmetice, geometric şi armonic unui grup de particule (395) măsurate cu micrometrul optico-electronic într-un eşantion de lichid (folosit la o instalaţie de acţionare de la o maşină-unealtă cu comandă numerică) s-au obţinut următoarele rezultate:

Numărul intervalului	Intervalul J	Diametrul, d_{nj}, mediu al intervalului	Nr. de particule n_j, din intervalul j
1	0– 4,9	2,5	10
2	5– 9,9	7,5	15
3	10–14,9	12,5	52
4	15–19,9	17,5	85
5	20–24,9	22,5	81
6	25–29,9	27,5	51
7	30–34,9	32,5	36
8	35–39,9	37,5	27
9	40–44,9	42,5	17
10	45–49,9	47,5	10
11	50–54,9	52,5	7
12	55–59,9	57,5	4

Să se calculeze:

- diametrul mediu aritmetic, $d_{n,a}$;
- diametrul mediu geometric, $d_{n,g}$;
- diametrul mediu armonic, $d_{n,ar}$.

Să se verifice relația:

$$d_{n,ar} < d_{n,g} < d_{n,a}$$

În caz de eroare, se va afişa mesajul "Relația $d_{n,ar} < d_{n,g} < d_{n,a}$ NU ESTE ÎNDEPLINITĂ".

Indicație. Pentru determinarea celor trei diametre se vor utiliza [ ] următoarele relații de calcul:

$$d_{n,a} = \frac{n_1 d_{n1} + n_2 d_{n2} + \ldots + n_n d_{nr}}{n_1 + n_2 + \ldots + n_r} = \frac{\sum_{j=1}^{r} (n_j d_{nj})_j}{\sum_{j=1}^{r} n_j}$$

$$d_{n,g} = \sqrt[r]{d_{n1} \cdot d_{n2} \cdot \ldots d_{nr}} = \sqrt[r]{\prod_{j=1}^{r} d_{nj}}$$

$$d_{n,ar} = \sum_{j=1}^{r} n_j \sum_{j=1}^{r} \left(\frac{d_{nj}}{n_j} \right)_j$$

Observație. În ultimele trei relații j reprezintă numărul curent al intervalului în care au fost grupate măsurătorile, în cazul nostru, r=12 intervale.

c) Să se verifice dacă triunghiul cu vîrfurile în punctele A(x1, y1), B(x2, y2), C(x3, y3) este dreptunghic.

d) Să se arate că patrulaterul cu vîrfurile în

$$M(1,5),\ N\left(\frac{2}{5}, \frac{19}{5}\right),\ P(2,3),\ Q\left(\frac{13}{5}, \frac{21}{5}\right)$$

este dreptunghi.

e) Să se verifice coliniaritatea a n puncte.

f) Să se verifice dacă punctele M(1,1), N(−5,5) şi P(0, $\frac{5}{3}$) sînt coliniare.

g) Să se determine punctul de intersecție a două drepte de ecuații:

$$a1x + b1y + c1 = 0$$

şi

$$a2x + b2y + c2 = 0$$

h) Să se determine punctul de intersecție a două mediane ale unui triunghi.

i) Să se scrie ecuația perpendicularei care trece prin mijlocul dreptei determinate de punctele A şi B de coordonate x1, y1 respectiv x2, y2.

j) Să se scrie ecuația dreptei care trece prin punctul $M(x_0, y_0)$ și este perpendiculară pe dreapta determinată de punctele A și B de coordonate x_1, y_1 respectiv x_2, y_2.

k) Să se determine distanța de la un punct la o dreaptă de ecuație $ax+by+c=0$.

l) Să se scrie ecuația dreptei determinate de punctele $A(x_1, y_1)$ și $B(x_2, y_2)$. Discuție.

m) Să se scrie ecuația dreptei care conține punctul M (−1, 3) și care este perpendiculară pe dreapta $3x-5y=0$.

n) Fie punctele P(2, 81), M(0, −79), N(−159,3), Q(161,−1). Să se arate că dreptele PM și NQ sînt perpendiculare.

o) Să se verifice dacă triunghiul cu vîrfurile în punctele (1, 2), (5, −1), (6, 15) și (0, 0), (0, 3), (4, 0) este dreptunghic.

☐ **Și în cadrul acestei teme se reia problema din conversația precedentă, complicînd-o sub următoarele aspecte:**

- se citește de la tastatură numărul de articole (maximum 5);
- se memorează într-un vector, printr-o procedură de introducere statică a datelor, cantitățile lunare de jucării fabricate;
- pentru fiecare articol (tip jucărie) se calculează procentul cantității totale trimestriale față de cantitatea totală anuală;
- pentru fiecare articol se afișează: număr trimestru (I, II, III, IV), cantitatea totală trimestrială, procentul cantității trimestriale față de cantitatea anuală.

☐ **Se reia problema din conversația precedentă, complicînd-o după cum urmează:**

Pentru un număr variabil de articole se citesc de la tastatură consumul normat și consumul realizat. Se calculează pentru fiecare articol abaterea în procente față de consumul normat și se tipăresc: codul articolului (1, 2, . . . , m), proveniența (R.S.R. sau import) și abaterea în procente. Locul de fabricație a articolului se citește sub forma codificată: 1 pentru „import" și 2 pentru „R.S.R.", dar se va edita în clar (R.S.R. sau IMPORT).

SOLUȚIA TEMEI 5

☐ Nu răspundem.

☐ Nu răspundem.

☐ **Programele BASIC sînt:**

a)
```
10 DIM P(15), Q(15)
20 READ N
30 PRINT "P(X)=";
40 FOR I=N+1 TO 1 STEP -1
50    READ P(I)
60    PRINT P(I);
```

```
 70 NEXT I
 80 REM ----------
 90 PRINT
100 PRINT "IMPĂRȚIT la X-";
110 INPUT X0
120 PRINT "CÎTUL";
130 P3=Q(N+1)=P(N+1)
140 FOR I=N TO 1 STEP-1
150   Q(I)=P3=P3*X0+P(I)
160 NEXT I
170 REM -----------
180 REM AFIȘARE REZULTAT
190 FOR I=N+1 TO 2 STEP-1
200       PRINT Q(I)
210 NEXT I
220 REM ---------
230 PRINT "RESTUL="; Q(1)
240 DATA 4
250 DATA 1, 2, -13, -14, 24.
260 END.
RUN
P(X)=1 2 -13 -14 24
ÎMPĂRȚIT LA X-?+1
CÎTUL 1 3-10-24 RESTUL=0
```

c)

```
 10 T=0
 20 WHILE T< >9999
 30 FOR I=1 TO 3
 40   READ X(I), Y(I)
 50   PRINT "("; X(I); ", ";
      Y(I); "),";
 60 NEXT I
 70 FOR I=1 TO 3
 80   X(I+3)=X(I)
 90   Y(I+3)=Y(I)
100 NEXT I
110 FOR I=1 TO 3
120   A=X(I)-X(I+1)
130   B=Y(I)-Y(I+1)
140   D(I)=SQR (A*A+B*B)
150   D(I+3)=D(I)
160 NEXT I
170 FOR I=1 TO 3
180     R=D(I) ↑ 2
190     S=D(I+1) ↑ 2+D(I+
        +2) ↑ 2
200     IF R=S THEN 240
210 NEXT I
220 PRINT "TRIUNGHIUL NU
    ESTE DREPTUNGHIC"
230 GOTO 260
240 PRINT "TRIUNGHIUL ESTE
    DREPTUNGHIC"
250 PRINT "IPOTENUZA ESTE
    DETERMINATĂ DE";
    X (I+1); Y(I+1)
260 READ T
270 WEND
280 DATA 1, 2, 5, -1, 6, 15, 9
290 DATA 0, 0, 0, 3, 4, 0, 9 9 9 9
300 END
```

e)

```
 10 INPUT "N="; N
 20 DIM X(N), Y(N)
 30 FOR I=1 TO N
 40   PRINT "X("; I; ")=";
 50   INPUT X(I)
 60   PRINT "Y("; I; ")=";
 70   INPUT Y(I)
 80 NEXT I
 90 M1=(Y(2)-Y(1))/(X(2)-X(1))
100 FOR I=2 TO N-1
110   M2=(Y(I+1)-Y(I))/(X(I+
      +1)-X(I)
120   IF M1< >M2 THEN 170
130   M1=M2
140 NEXT I
150 PRINT N; "PUNCTE COLINIARE"
160 STOP
170 PRINT "PUNCTE NECOLINIARE"
180 END

RUN
 N=?5
 X(1)=?1
 X(2)=?3
 X(3)=?5
 X(4)=?7
 X(5)=?9
 Y(1)=?2
 Y(2)=?4
 Y(3)=?6
 Y(4)=?8
 Y(5)=?10

5 PUNCTE COLINIARE

RUN
 N=?4
 X(1)=?1
 X(2)=?3
 X(3)=?2
 X(4)=?3
 Y(1)=?8
 Y(2)=?9
 Y(3)=?7
 Y(4)=?2

PUNCTE NECOLINIARE
```

g) Fiind date două drepte de ecuație:

$$a_1x+b_1y+c_1=0$$

și

$$a_2x+b_2y+c_2=0$$

în urma rezolvării sistemului de ecuaţii rezultă coordonatele (x, y) punctului de intersecţie.

$$x=\frac{b1\cdot c2-b2\cdot c1}{b2\cdot a1-b1\cdot a2},$$

şi

$$y=\frac{a2\cdot c1-a1\cdot c2}{a1\cdot b2-a2\cdot b1}.$$

Atunci cînd $b2\cdot a1-b1\cdot a2=0$, rezultă:

$$\frac{a1}{b1}=\frac{a2}{b2}, \text{ sau}$$

$-\frac{a1}{b1}=-\frac{a2}{b2}$; ce reprezintă condiţia de paralelism a două drepte $\left(y=-\frac{a1}{b1}x-\frac{c1}{b1}\right.$ şi $\left.y=-\frac{a2}{b2}\cdot x-\frac{c2}{b2}\right.$; cele două drepte au pantele $-\frac{a1}{b1}$ şi $-\frac{a2}{b2}$ egale).

Cînd b1 sau b2 sînt zero, rezultă

$$a1x+c1=0,\ x=-\frac{c1}{a1},$$

şi

$$a2x+c2=0,\ x=-\frac{c2}{a2},$$

```
10 READ A1, B1, C1
20 WHILE A1<>999
30   READ A2, B2, C2
40   PRINT A1; "*X+("; B1; ")
     *Y+(";C1;")=0"
50   PRINT A2;"*X+(";B2;")
     *Y+(";C2;")=0"
60   D=A1*B2-A2*B1
70   IF D<>0 THEN
80     N1=A2*C1-A1*C2
90     N2=B1*C2-B2*C1
100    X=N2/D
110    Y=N1/D
120    PRINT "SOLUŢIA ESTE:
       ("; X; ", "; Y; ")"
130  ELSE
140  IF B1=0 THEN
150    IF C1/A1=C2/A2 THEN
160      PRINT "DREPTELE
         COINCID"
170    ELSE
180      PRINT "DREPTE
         PARALELE"
190    ENDIF
200  ELSE
210    IF C1/B1=C2/B2 THEN
220      PRINT "DREPTELE
         COINCID"
230    ELSE
240      PRINT "DREPTE
         PARALELE"
250    ENDIF
260  ENDIF
270  ENDIF
280  READ A1, B1, C1
290 WEND
300  DATA 1, 1, 1, 1, 2, 2, 2
310  DATA 1, 1, 2, 3, 4, 5, 6
320  DATA 1, 5, 3, 10, 10, 6, -3
330  DATA 1, 3, 0, 4, 7, 0, -8
340  DATA 999, 0, 0
350 END
```

i)

```
10 READ X1, Y1
20 WHILE X1<>999
30   READ X2, Y2
40   PRINT X1; Y1; X2; Y2
50   XO=(X1+X2)/2
60   YO=(Y1+Y2)/2
70   IF (X1-X2)<>0
80     IF (Y1-Y2)<>0 THEN
90       M=(Y2-Y1)/(X2-X1)
100      MO=-1/M
110      N=YO-MO * XO
120      PRINT MO, N
130    ELSE
```

```
140       PRINT "X="; XO
150     ENDIF
160   ELSE
170     IF (Y1-Y2)<>0 THEN
180       PRINT "Y="; YO
190     ELSE
200       PRINT "PUNCTELE
          COINCID"
210     ENDIF
220   ENDIF
230   READ X1, Y1
235 WEND
240 DATA 1, 2
250 DATA 4, 7, 4, 7, 4, 3
260 DATA -3, 9, 4, 5, 4, -7, 4,
    -7, 999, 0
270 END
```

j)

```
10 INPUT "XO, YO"; XO, YO
30 WHILE XO<>999
35   INPUT "X1, Y1"; X1, Y1
40   INPUT "X2, Y2"; X2, Y2
50   V=Y2-Y1
60   H=X2-X1
70   IF V<>0 THEN
80     IF H<>0 THEN
90       M=V/H
100      MO=-1/M
110      N=YO-MO * XO
120      PRINT MO, N
130    ELSE
140      PRINT "Y=", YO
150    ENDIF
160  ELSE
170    IF H<>0 THEN
180      PRINT "X=", XO
190    ELSE
200      PRINT "PUNCTELE
         COINCID"
210    ENDIF
220  ENDIF
230  INPUT "XO, YO"; XO, YO
240 WEND
250 END
```

☐ **Pentru scrierea programului s-au definit** variabilele de intrare – Z (număr articole), C (cantitățile lunare de jucări pentru un anumit tip de articol); variabilele de stare – S (cantitate totală anuală de jucării), L, I, K, R (variabile de lucru), C1 (cîtul împărțirii), R (restul împărțirii), T (cantitate totală trimestrială); variabila de ieșire – P (consumuri procentuale trimestriale).

```
5 REM PROGRAMUL CITEȘTE CANTITĂȚILE LUNARE (C) DE JUCĂRII PENTRU
10 REM UN NUMĂR (Z) VARIABIL DE ARTICOLE, CALCULEAZĂ
15 REM PROCENTUL (P) CANTITĂȚII TOTALE TRIMESTRIALE
20 REM FAȚĂ DE CANTITATEA TOTALĂ ANUALĂ (S) ȘI
25 REM TIPĂREȘTE REZULTATUL
30 PRINT "INDICAȚI NUMĂRUL DE ARTICOLE";
40 INPUT Z
50 DIM C(12), P(4)
60 PRINT "TRIMESTRUL 1 TRIMESTRUL 2 TRIMESTRUL 3 TRIMESTRUL 4"
70 PRINT
75 FOR K=1 TO Z
80   FOR I=1 TO 12
90     READ C (I)
100  NEXT I
110  DATA 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120
120  DATA 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121
130  DATA 12, 22, 32, 42, 52, 62, 72, 82, 92, 103, 112, 122
140  DATA 13, 23, 33, 43, 53, 63, 73, 83, 93, 102, 113, 123
150  DATA 14, 24, 34, 44, 54, 64, 74, 84, 94, 104, 114, 124
160  REM CALCUL CANTITATE TOTALĂ ANUALĂ JUCĂRII
170    S=0
180  FOR I=1 TO 12
190    S=S+C(I)
200  NEXT I
210 REM CALCUL PROCENTE
220 FOR L=1 TO 12
230   C1=INT(L/3)*3
240   R=L-C1
250   IF R<>1 THEN 270
260   T=0
270   T=T+C(L)
280   IF R<>0 THEN 300
```

```
290    P (C1/3)=T*100/S
300 NEXT L
310 REM EDITARE REZULTATE
320   FOR R=1 TO 4
330     PRINT P(R),
340   NEXT R
345   PRINT
350 NEXT K
```

Remarci

- Introduceți și executați programul aplicației.
- În momentul cînd noi am introdus pentru prima dată programul, am tastat linia 290, de maniera

```
290 P(C1)=T*100/S
```

Analizați singuri implicațiile datorate tastării greșite a acestei instrucțiuni, simulînd cu creionul și hîrtia modul de funcționare a procedurii de calcul procente din liniile 210–300. Atenție la valorile lui C1 și P!
Instrucțiunea (corectă)

```
290 P(C1/3)=T*100/S
```

mai poate fi scrisă și în alt mod? Poate fi înlocuită cu instrucțiunea

```
290 P(L/3)=T*100/S
```

Nu vă grăbiți cu răspunsul. Dacă aveți dificultăți apelați la prietenul dvs. calculatorul.

Observație. Rețineți procedura de calcul procente.

```
210 REM CALCUL PROCENTE
220 FOR L=1 TO 12
230    C1=INT (L/3)*3
240    R=L-C1
250    IF R<>1 THEN 270
260    T=0
270    T=T+C(L)
280    IF R<>0 THEN 300
290    P (C1/3)=T*100/S
300 NEXT L
```

□ **Pentru scrierea programului s-au definit** variabilele de intrare – M (numărul de articole), N (consumul normat), F (codul de fabricație), R (consumul realizat) și variabila de ieșire A (abaterea în procente față de consumul normat).

```
  5 REM PROGRAMUL CITEȘTE CONSUMUL NORMAT (N)
 10 REM ȘI CONSUMUL REALIZAT (R) PENTRU UN
 15 REM NUMĂR VARIABIL DE ARTICOLE (M)
 20 REM PROGRAMUL CALCULEAZĂ ABATEREA
 25 REM ÎN PROCENTE (A) FAȚĂ DE CONSUMUL NORMAT
 50 REM ******************************
 60 INPUT "NUMĂR ARTICOLE"; M
 70 FOR L=1 TO M
 80   PRINT "CONSUMUL NORMAT PENTRU ARTICOLUL:"; I;
 90   INPUT N
100   PRINT
110   PRINT "CODUL DE FABRICAȚIE 1 (IMPORT)/2(R.S.R.)";
120   INPUT F
130   PRINT
140   PRINT "CONSUMUL REALIZAT";
150   INPUT R
160   PRINT
170   A=(R-N)*100/N
```

```
180 IF F=1 THEN 200
190 PRINT "ARTICOLUL", I, "R.S.R.", "ABATEREA=", A, "%"
195 GOTO 210
200 PRINT "ARTICOLUL"; I, "IMPORT", "ABATEREA="; A; "%"
210 NEXT I
220 END

RUN

NUMĂR ARTICOLE ? 3
CONSUMUL NORMAT PENTRU ARTICOLUL : 1 ? 30
CODUL DE FABRICAȚIE 1 (IMPORT)/2 (R.S.R.) ? 1
CONSUMUL REALIZAT ? 15
ARTICOLUL 1 IMPORT ABATEREA=-50%
.
.
.
```

Remarci

- Introduceți și executați programul aplicației.
- În momentul introducerii programului nu uitați să tastați și linia 195. În absența liniei 195 s-ar fi întîmplat ceva grav?
- Rescrieți secvența de instrucțiuni din liniile 190–200 uitlizînd facilitățile limbajului GW-BASIC.

CONVERSAȚIA 6

Proiectarea și realizarea unui program pentru editarea raportului săptăminal privind livrările zilnice de lichide. Variabile indexate alfanumerice. Instrucțiuni matriciale. Funcțiile DEF FN și RANDOMIZE. Instrucțiunea PRINT AT. Programe utile pentru matematică și fizică. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLELE 6 – PC, m, M, F

☐ De la problemă la program

Vom aborda şi în cadrul acestei conversaţii aceeaşi problemă, complicînd-o după cum urmează.

Se citesc de la tastatură livrările de benzină efectuate zilnic (luni, marţi, miercuri, joi, vineri, sîmbătă) din trei rezervoare (cilindrice echilaterale) R1, R2, R3. Se doreşte ca, pentru fiecare rezervor în parte, să se afişeze, pe zile, cantităţile de benzină livrate beneficiarilor. Se va tipări, de asemenea, media livrărilor pe zile şi pe rezervoare. La finele listei se va afişa maximul livrărilor, precizîndu-se atît rezervorul cît şi ziua respectivă.

☐ Analiza problemei

Formatul datelor de ieşire, datele de intrare, tabela de variabile, specificaţiile de programare şi alocarea funcţiunilor de prelucrare sînt ilustrate în modulul de analiză structurată, fig. 6.1.

FORMATUL DATELOR DE IEŞIRE

Nume program: EXEMPLELE 6 – PC, m, M, F

ZIUA	R1	R2	R3	MEDIA
Luni	**	**	**	**.**
Marţi	**	**	**	**.**
Miercuri	**	**	**	**.**
Joi	**	**	**	**.**
Vineri	**	**	**	**.**
Sîmbătă	**	**	**	**.**
MEDIA	**.**	**.**	**.**	

MAXIMUL LIVRARILOR ** TONE
LA REZERVORUL = **
IN ZIUA DE = ******

a)

Fig. 6.1. a-e. Modulul de analiză structurată: a) formatul datelor de ieşire;

DATE INTRARE

Nume program: EXEMPLELE 6 – PC, m, M, F

INTRODUCEȚI LIVRĂRILE DE BENZINĂ
PE ZILE ÎN ORDINEA
R1 R2 R3

LUNI:
REZERVORUL 1 ? 15
REZERVORUL 2 ? 18
REZERVORUL 3 ? 30

MARȚI:
REZERVORUL 1 ? 23
REZERVORUL 2 ? 40
REZERVORUL 3 ? 0

MIERCURI:
REZERVORUL 1 ? 55
REZERVORUL 2 ? 26
REZERVORUL 3 ? 1

JOI:
REZERVORUL 1 ? 0
REZERVORUL 2 ? 0
REZERVORUL 3 ? 25

VINERI:
REZERVORUL 1 ? 12
REZERVORUL 2 ? 60
REZERVORUL 3 ? 60

SÎMBĂTĂ:
REZERVORUL 1 ? 60
REZERVORUL 2 ? 60
REZERVORUL 3 ? 60

b)

TABELA DE VARIABILE

Nume program: EXEMPLELE 6 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
A: matricea livrărilor zilnice de benzină	C$: vectorul zilelor săptămînii L: index coloană (aMIC) I: index Z: index S: total general livrări (zi/rezervor) M: index coloană (aMIC) C: numărul rezervorului R: ziua maximului livrărilor P: număr linie (aMIC); (HC-85) Q: număr coloană (aMIC); (HC-85)	B: vectorul livrărilor medii zilnice D: vectorul livrării medii pe rezervor H: maximul livrărilor C: numărul rezervorului R: ziua maximului livrărilor

c)

b) datele de intare; *c)* tabela de variabile;

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 6 – PC, m, M, F

Descrierea programului. Programul editează situaţia livrărilor de benzină efectuate zilnic din trei rezervoare cilindrice echilaterale.

Livrările, pe zile şi pentru fiecare rezervor în parte se citesc de la tastatură. Programul mai afişează maximul livrărilor insoţit de rezervorul şi ziua respectivă.

Intrări. Valorile livrărilor, pe zile, pentru fiecare rezervor.

Ieşiri. Lista cu situaţia livrărilor.

Lista de funcţiuni ale programului

1. Citeşte zilele săptămînii.
2. Citeşte livrările zilnice pentru fiecare din cele trei rezervoare.
3. Însumează livrările pentru fiecare rezervor.
4. Calculează media livrărilor pe zile.
5. Iniţializează variabila S.
6. Însumează livrările în total general livrări rezervor.
7. Calculează media livrărilor pe rezervor.
8. Afişează ziua, livrările din rezervoare şi media livrărilor pe zi.
9. Afişează un rînd de " – ".
10. Afişează media livrărilor pe rezervor.
11. Calculează maximul livrărilor.
12. Iniţializează variabilele R, C şi H.
13. Afişează maximul livrărilor, numărul rezervorului şi ziua.
14. Afişează cap tabel.
15. Stop.
16. Calculează P (aMIC).
17. Calculează Q (aMIC).
18. Şterge ecranul.

d)

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 6 – PC, m, M, F

Modul	Funcţiuni
CAP–TAB	14, 18 (aMIC, PRAE)
CIT–ZILE	1
CIT–LIV	2
SUM–LIV	5, 3
MED–LIV	4
SUM–REZ	5, 6
MED–REZ	7
EDIT–RIND	8, 9, 16, 17 (aMIC, HC-85)
EDIT–COL	10, 17 (aMIC, HC-85)
CALC–MAX	11, 12
EDIT–MAX	13
FINAL	15

e)

d) specificaţii de programare; *e)* alocarea funcţiunilor de prelucrare.

Ne vom opri, în cele ce urmează, asupra tabelei de variabile în care s-au definit două noi tipuri de variabile – variabila structurată (de intrare) de tip matrice (A) şi variabila structurată alfanumerică de tip vector C$ (de stare).

Variabile structurate de tip tablou (matrice)

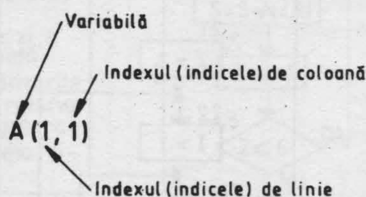
Tablourile (matricile sau masivele) sînt un ansamblu de elemente dispuse pe linii şi coloane. Prelucrările ce se efectuează asupra elementelor unei matrice sînt funcţie de valorile unor indici de poziţie, ce precizează coordonatele (poziţia) liniei şi ale coloanei corespunzătoare. Pentru EXEMPLELE 6 – PC, m, M, F s-a definit variabila structurată de tip matrice (tablou) A în care se citesc cantităţile de benzină livrate zilnic din fiecare rezervor. Tabloul conţine 6 linii (Z) şi 3 coloane (I) pentru livrările zilnice din cele trei rezervoare.

o I → Rezervoare

Z ↓ Zile

Z \ I	1	2	3
1	15	18	30
2	23	40	0
3	55	26	1
4	0	0	25
5	12	60	60
6	60	60	60

Observaţie. **Z şi I sînt definite ca două variabile de stare, reprezentînd indexul (indicele) de linie, respectiv indexul (indicele) de coloană.**



Notă : Se citeşte "A de unu, unu"

Variabila de stare C$ (şir sau alfanumerică) a fost introdusă pentru definirea vectorului în care urmează să se citească, în mod static zilele săptămînii.

C$(1)	C$(2)	C$(3)	C$(4)	C$(5)	C$(6)
LUNI	MARŢI	MIERCURI	JOI	VINERI	SÎMBĂTĂ

□ Proiectarea programului

Diagrama de structură, schema logică şi pseudocodul sînt prezentate în modulul de proiectare structurată, fig. 6.2.

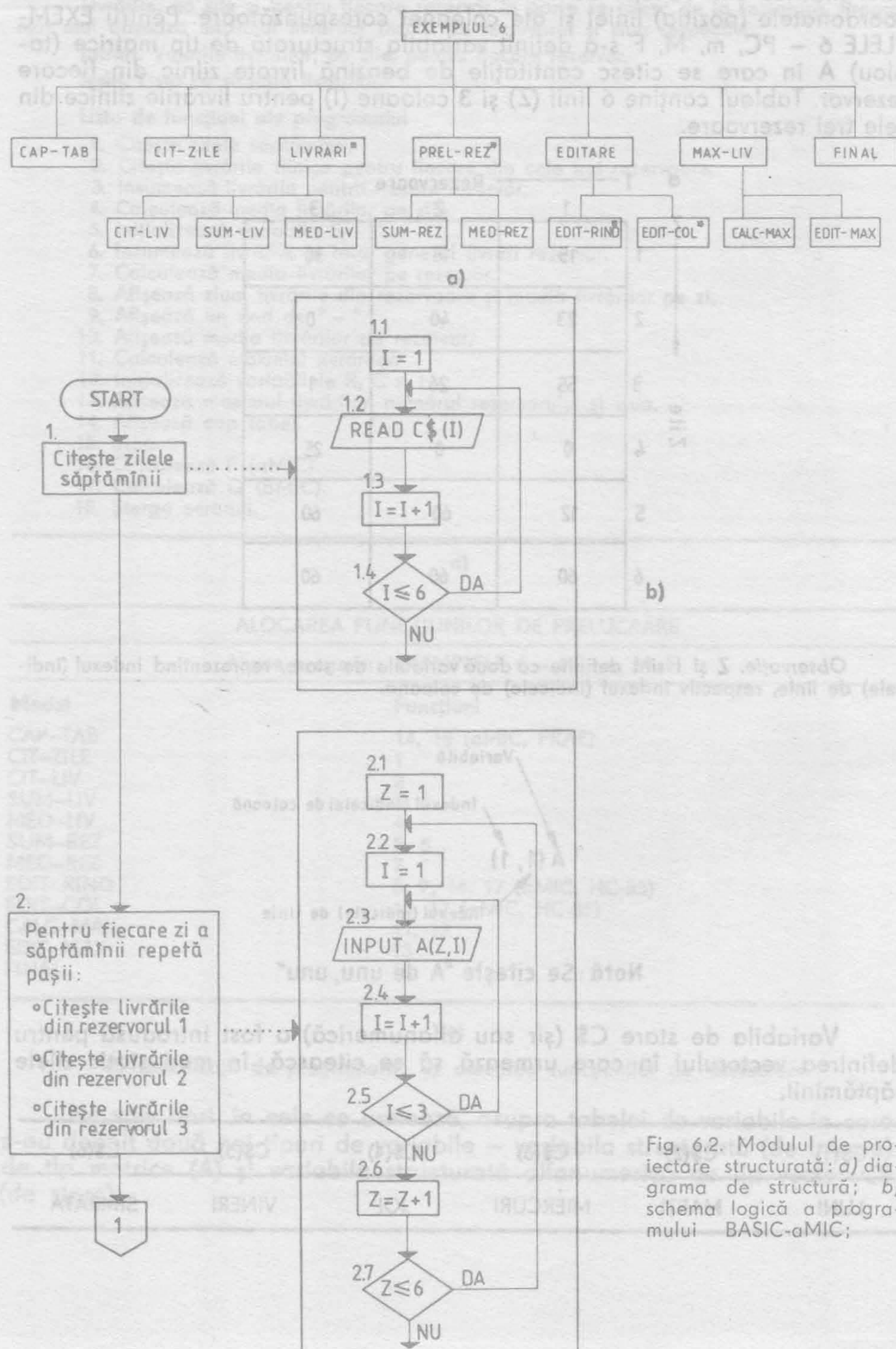
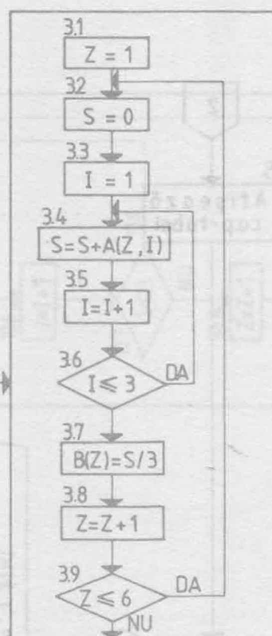


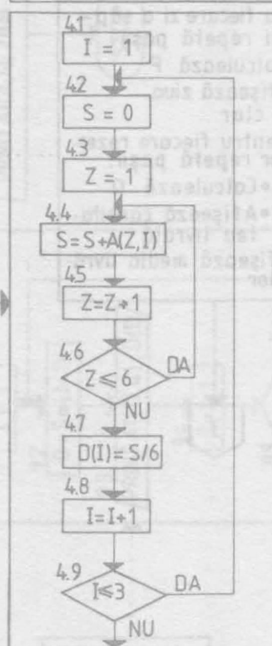
Fig. 6.2. Modulul de proiectare structurată: *a)* diagrama de structură; *b)* schema logică a programului BASIC-aMIC;

1

3. Pentru fiecare zi a săptămînii repetă pașii:
- Inițializează pe S cu zero
- Pentru fiecare rezervor repetă:
  - Însumează livrările din fiecare rezervor în totalul general
- Calculează media livrărilor



4. Pentru fiecare rezervor repetă pașii:
- Inițializează pe S cu zero
- Pentru fiecare zi a săptămînii repetă:
  - Însumează livrările din fiecare rezervor în totalul general
- Calculează media livrărilor



2

Fig. 6.2(b).

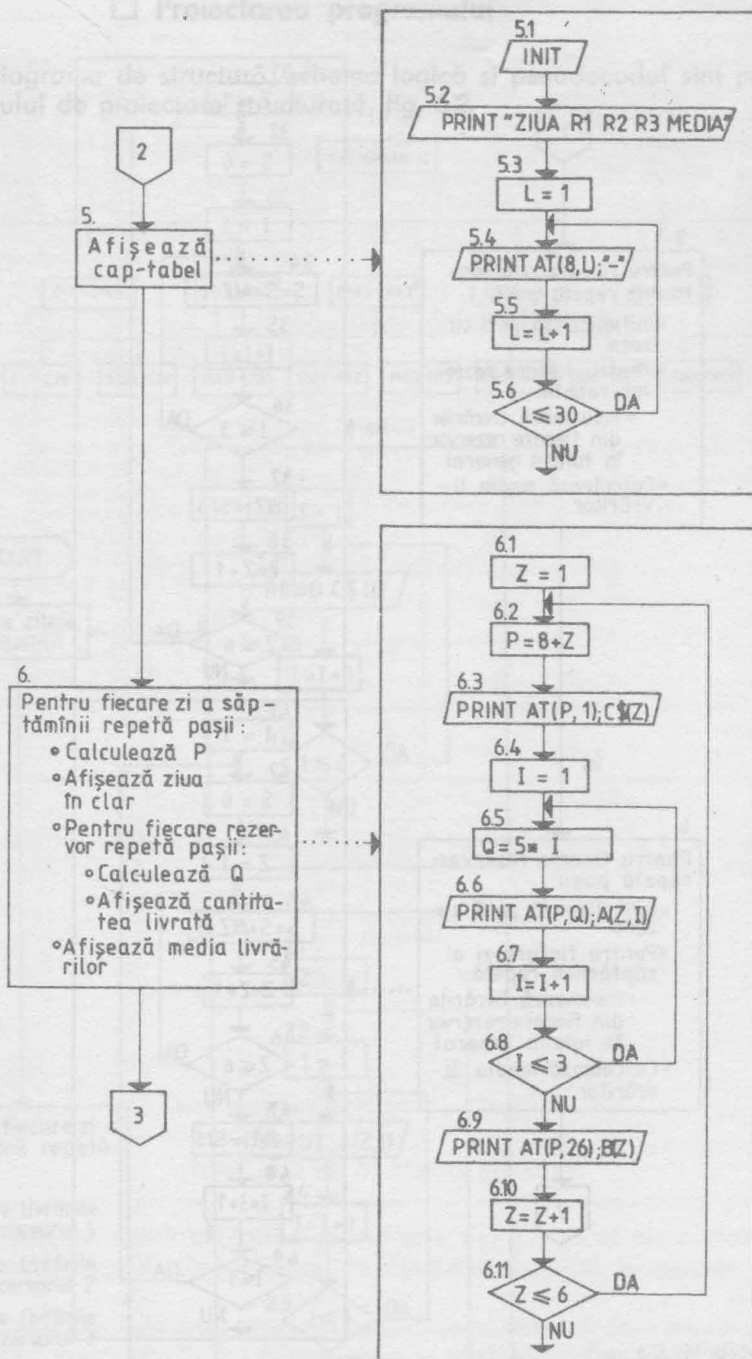

2
5.
Afişează cap-tabel
5.1
INIT
5.2
PRINT "ZIUA R1 R2 R3 MEDIA"
5.3
L = 1
5.4
PRINT AT(8,L);"-"
5.5
L = L+1
5.6
L ≤ 30
DA
NU
6.
Pentru fiecare zi a săptămînii repetă paşii:
• Calculează P
• Afişează ziua în clar
• Pentru fiecare rezervor repetă paşii:
• Calculează Q
• Afişează cantitatea livrată
• Afişează media livrărilor
6.1
Z = 1
6.2
P = 8+Z
6.3
PRINT AT(P, 1); C$(Z)
6.4
I = 1
6.5
Q = 5* I
6.6
PRINT AT(P,Q); A(Z,I)
6.7
I = I+1
6.8
I ≤ 3
DA
NU
6.9
PRINT AT(P,26); B(Z)
6.10
Z = Z+1
6.11
Z ≤ 6
DA
NU
3

Fig. 6.2(b).

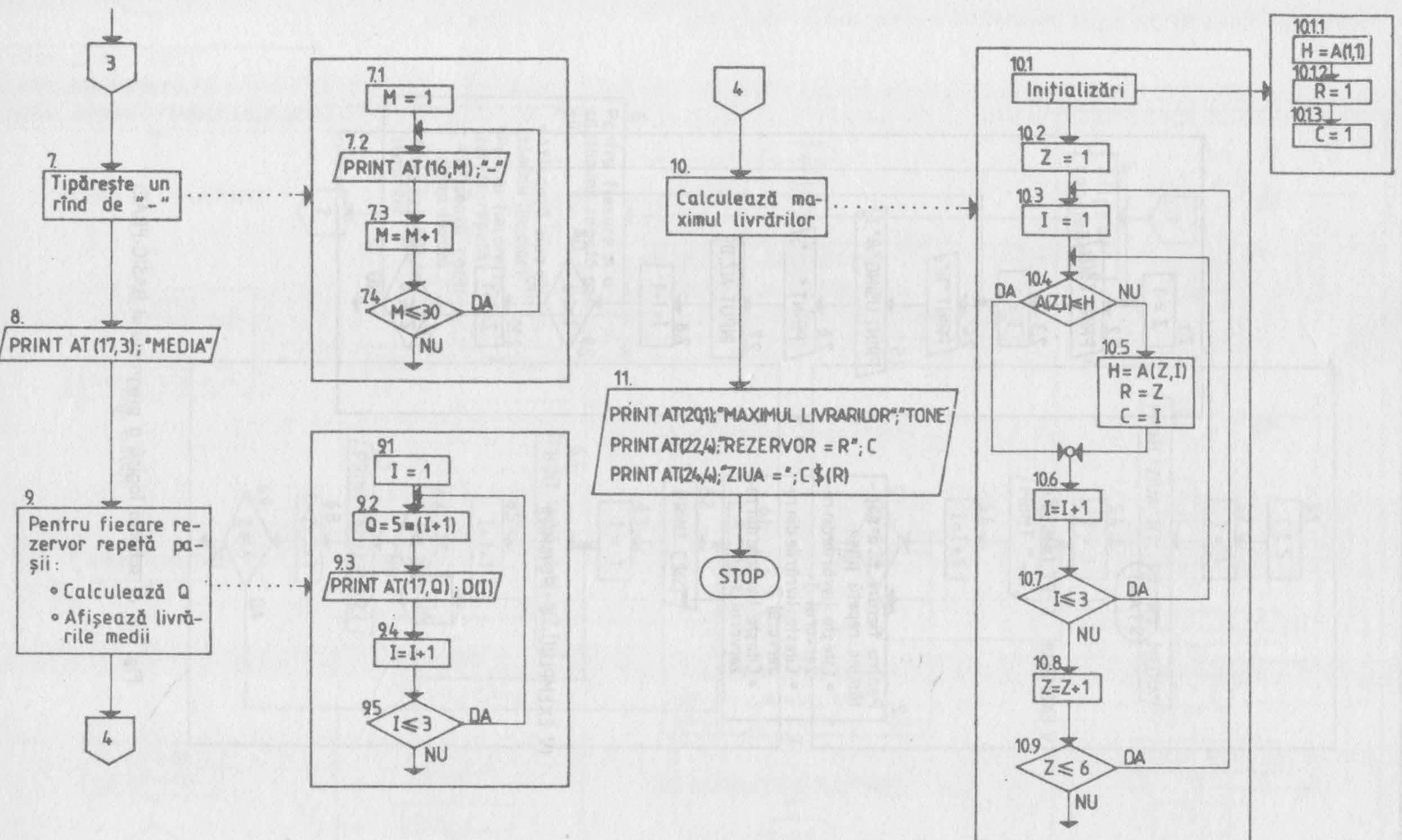

3
7. Tipăreşte un rînd de "–"
7.1 M = 1
7.2 PRINT AT(16,M);"–"
7.3 M = M+1
7.4 M≤30
DA
NU
8. PRINT AT(17,3);"MEDIA"
9. Pentru fiecare rezervor repetă paşii:
• Calculează Q
• Afişează livrările medii
9.1 I = 1
9.2 Q = 5*(I+1)
9.3 PRINT AT(17,Q);D(I)
9.4 I=I+1
9.5 I≤3
DA
NU
4
4
10. Calculează maximul livrărilor
10.1 Iniţializări
10.1.1 H = A(1,1)
10.1.2 R = 1
10.1.3 C = 1
10.2 Z = 1
10.3 I = 1
10.4 A(Z,I)≤H
DA
NU
10.5 H=A(Z,I) R = Z C = I
10.6 I=I+1
10.7 I≤3
10.8 Z=Z+1
10.9 Z≤6
11. PRINT AT(20,1);"MAXIMUL LIVRARILOR";"TONE"
PRINT AT(22,4);"REZERVOR = R";C
PRINT AT(24,4);"ZIUA = ";C$(R)
STOP

Fig. 6.2(b).

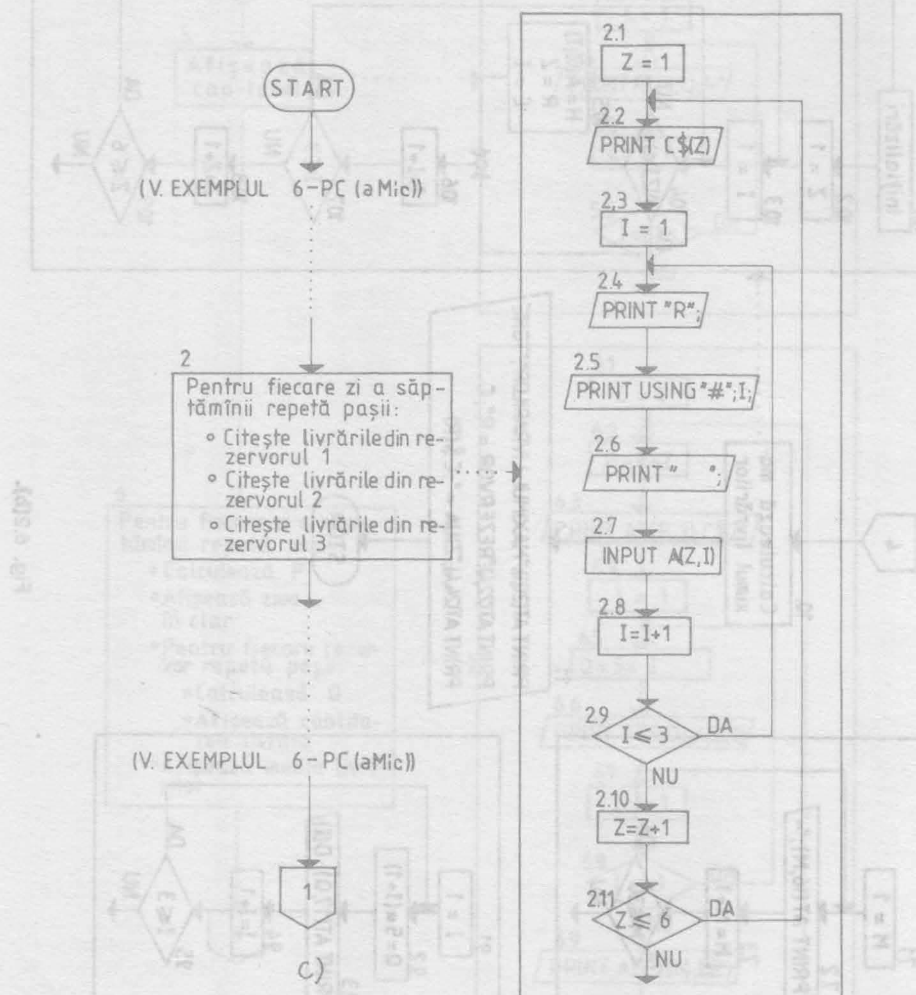


Fig. 6.2. c) schema logică a programului BASIC-PRAE;

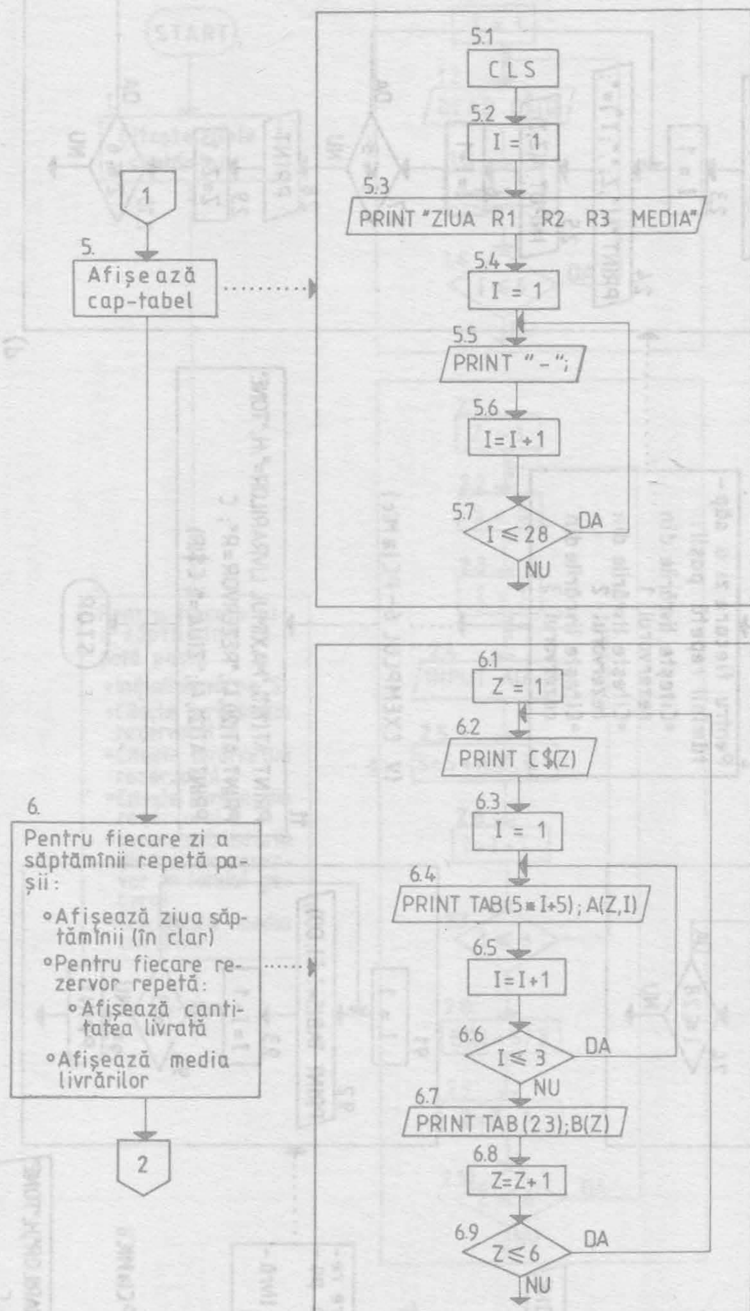

1
5.
Afişează cap-tabel
5.1
CLS
5.2
I = 1
5.3
PRINT "ZIUA R1 R2 R3 MEDIA"
5.4
I = 1
5.5
PRINT "-";
5.6
I=I+1
5.7
I ≤ 28
DA
NU
6.
Pentru fiecare zi a săptămînii repetă paşii:
∘Afişează ziua săptămînii (în clar)
∘Pentru fiecare rezervor repetă:
∘Afişează cantitatea livrată
∘Afişează media livrărilor
2
6.1
Z = 1
6.2
PRINT C$(Z)
6.3
I = 1
6.4
PRINT TAB(5*I+5); A(Z,I)
6.5
I=I+1
6.6
I ≤ 3
DA
NU
6.7
PRINT TAB(23);B(Z)
6.8
Z=Z+1
6.9
Z≤6
DA
NU

Fig. 6.2(c).

2

7\. Tipărește un rînd de "-"

7.1 I = 1
7.2 PRINT "-";
7.3 I=I+1
7.4 I≤28 — DA / NU

8\. PRINT "MEDIA"

9\. Pentru fiecare rezervor repetă pasul:
- Afișează livrările medii

9.1 I = 1
9.2 PRINT TAB(5*I+5); D(I);
9.3 I=I+1
9.4 I≤3 — DA / NU
9.5 PRINT

(V. EXEMPLUL 6-PC(a MIC))

11\.
```
PRINT "MAXIMUL LIVRARILOR";H;"TONE"
PRINT "REZERVOR = R"; C
PRINT "ZIUA = "; C$(C)
```

STOP

Fig. 6.2(c).

START

(V. EXEMPLUL 6-PC(a Mic)

2\. Pentru fiecare zi a săptămînii repetă pașii:
- Citește livrările din rezervorul 1
- Citește livrările din rezervorul 2
- Citește livrările din rezervorul 3

2.1 Z = 1
2.2 PRINT C$(Z);":";
2.3 I = 1
2.4 PRINT "A(";Z;",";I")=";
2.5 INPUT A(Z,I)
2.6 I=I+1
2.7 I≤3 — DA / NU
2.8 PRINT
2.9 Z=Z+1
2.10 Z≤6 — DA / NU

d)

(V. EXEMPLUL 6-PC(a Mic)

11\.
```
PRINT AT(19,1);"MAXIMUL LIVRARILOR=";H;"TONE"
PRINT AT(20,4);"REZERVOR=R"; C
PRINT AT(21,4);"ZIUA="; C$(R)
```

STOP

Fig. 6.2(d). schema logică a programului BASIC HC-85, TIM S, SPECTRUM;

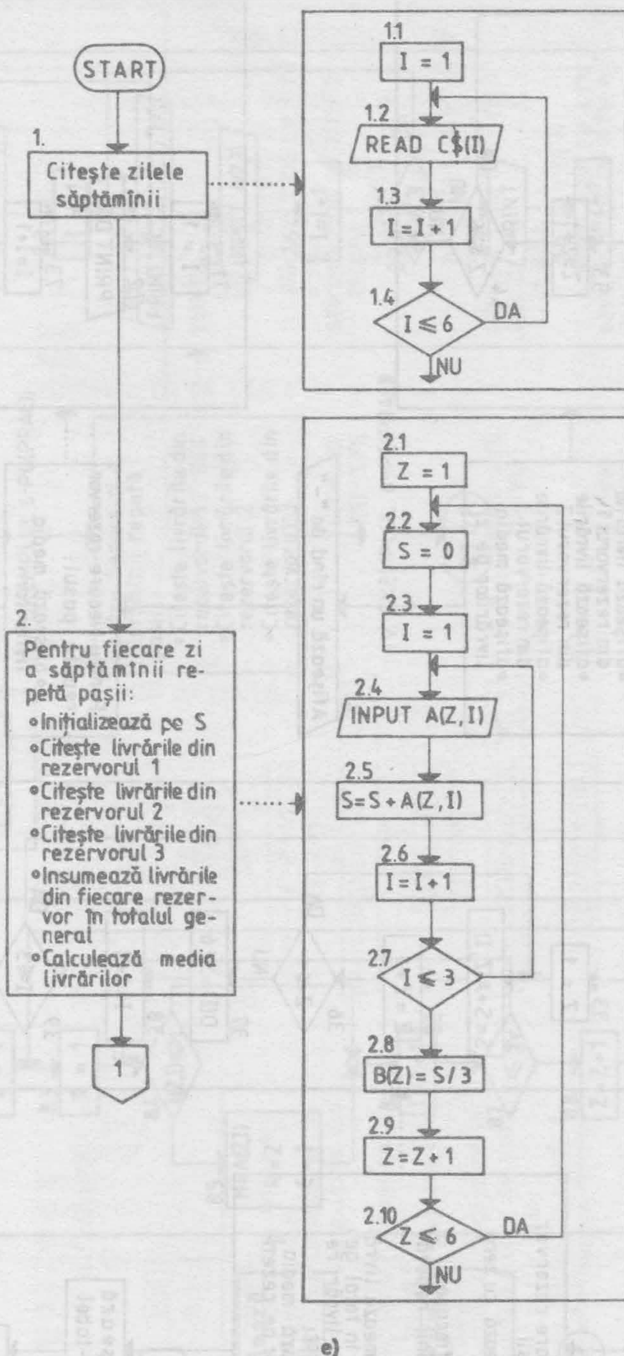


Fig. 6.2. e) schema logică a programului BASIC-AMSTRAD, BASIC-80, ABASIC.

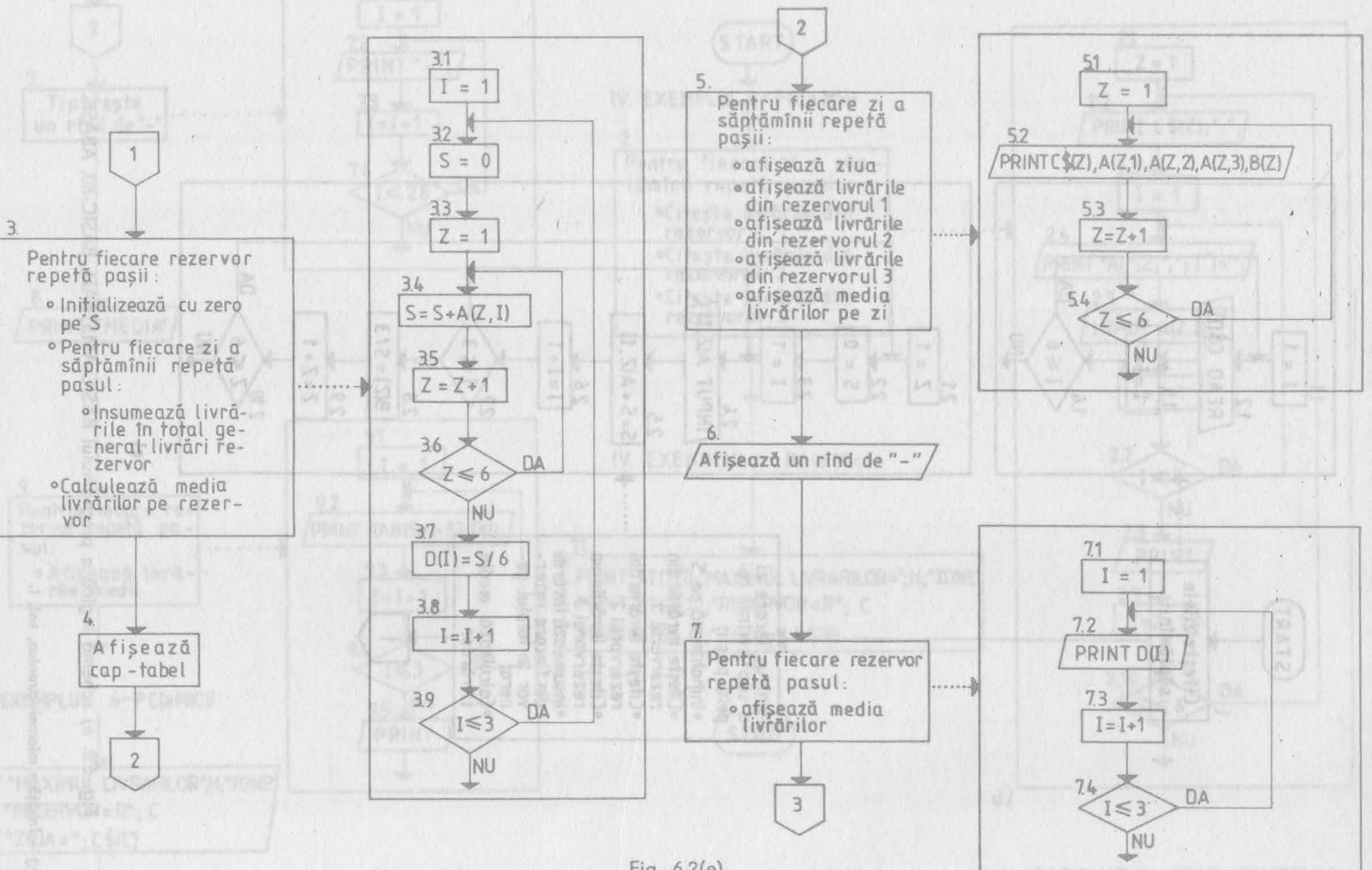

1
3.
Pentru fiecare rezervor repetă pașii:
• Inițializează cu zero pe S
• Pentru fiecare zi a săptămînii repetă pasul:
• Însumează livrările în total general livrări rezervor
• Calculează media livrărilor pe rezervor
4.
Afișează cap-tabel
2
3.1
I = 1
3.2
S = 0
3.3
Z = 1
3.4
S = S+A(Z,I)
3.5
Z = Z+1
3.6
Z ≤ 6
DA
NU
3.7
D(I) = S/6
3.8
I = I+1
3.9
I ≤ 3
DA
NU
2
5.
Pentru fiecare zi a săptămînii repetă pașii:
• afișează ziua
• afișează livrările din rezervorul 1
• afișează livrările din rezervorul 2
• afișează livrările din rezervorul 3
• afișează media livrărilor pe zi
6.
Afișează un rînd de "-"
7.
Pentru fiecare rezervor repetă pasul:
• afișează media livrărilor
3
5.1
Z = 1
5.2
PRINT C$(Z),A(Z,1),A(Z,2),A(Z,3),B(Z)
5.3
Z=Z+1
5.4
Z ≤ 6
DA
NU
7.1
I = 1
7.2
PRINT D(I)
7.3
I=I+1
7.4
I ≤ 3
DA
NU

Fig. 6.2(e).

e)

Fig. 6.2(e).

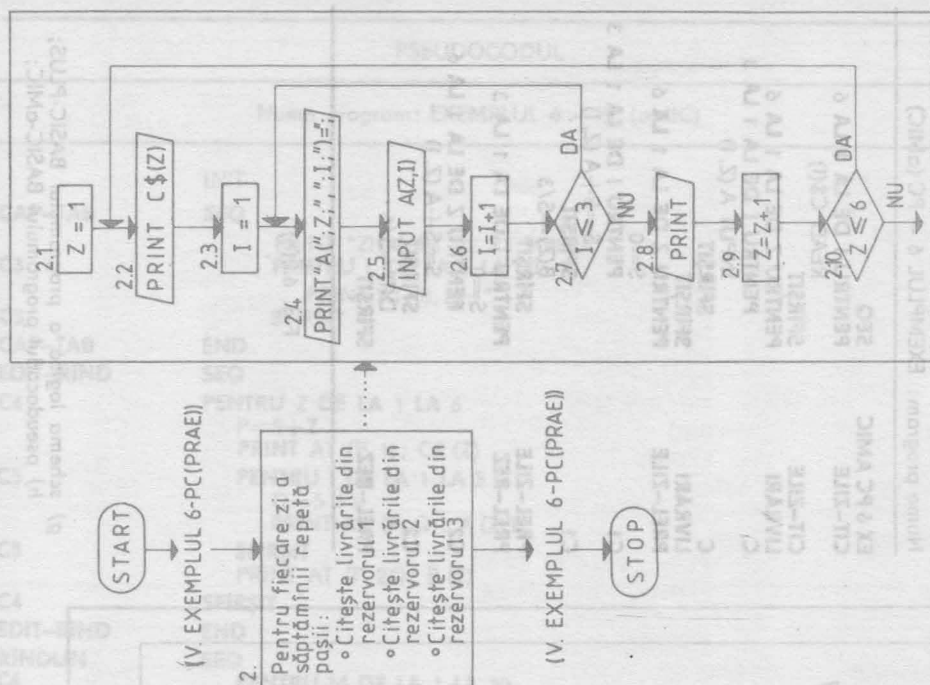


Fig. 6.2(f). f) schema logică a programului BASIC-COMMODORE;

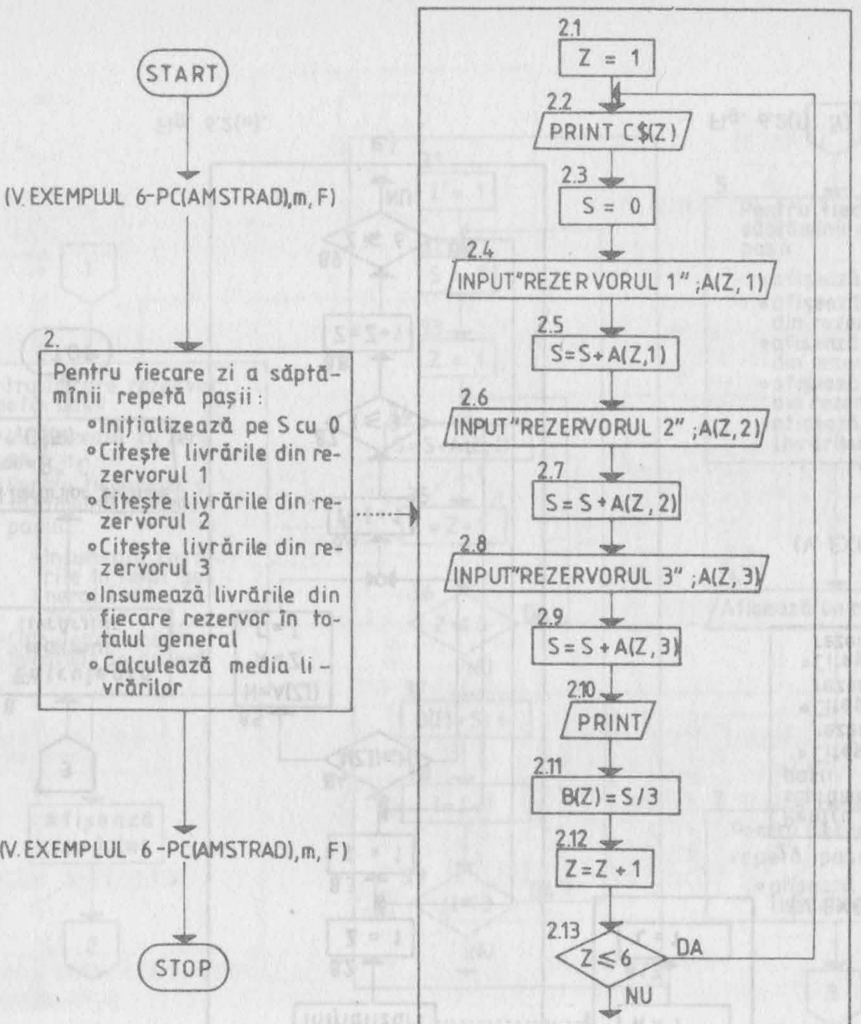


Fig. 6.2(g).

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (aMIC)

```
EX 6 PC AMIC    SEQ
CIT–ZILE        PENTRU I DE LA 1 LA 6
                     READ C$(I)
CIT–ZILE        SFIRȘIT
LIVRĂRI         PENTRU Z DE LA 1 LA 6
C                    PENTRU I DE LA 1 LA 3
                          INPUT A (Z, I)
C                    SFIRȘIT
LIVRĂRI         SFIRȘIT
PREL–ZILE       PENTRU Z DE LA 1 LA 6
                          S=0
C1                        PENTRU I DE LA 1 LA 3
                               S=S+A (Z, I)
C1                        SFIRȘIT
                          B(Z)=S/3
PREL–ZILE            SFIRȘIT
PREL–REZ        PENTRU I DE LA 1 LA 3
                     S=0
C2                   PENTRU Z DE LA 1 LA 6
                          S=S+A (Z, I)
C2                   SFIRȘIT
                     D(I)=S/6
PREL–REZ        SFIRȘIT
```

Fig. 6.2(h).

g) schema logică a programului BASIC-PLUS;
h) pseudocodul programului BASIC-aMIC;

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (aMIC)

```
                INIT
CAP-TAB         SEQ
                     PRINT "ZIUA R1 R2 R3 MEDIA"
C3                   PENTRU L DE LA 1 LA 30
                        PRINT AT (8, L); " – "
C3                   SFÎRŞIT
CAP-TAB         END
EDIT-RÎND       SEQ
C4              PENTRU Z DE LA 1 LA 6
                     P=8+Z
                     PRINT AT (P, 1); C$ (Z)
C5                   PENTRU I DE LA 1 LA 3
                        Q=5 * I
                        PRINT AT (P, Q); A (Z, I)
C5                   SFÎRŞIT
                     PRINT AT (P, 26); B (Z)
C4              SFÎRŞIT
EDIT-RÎND       END
RÎNDLIN         SEQ
C6                   PENTRU M DE LA 1 LA 30
                        PRINT AT (16, M); " – "
C6                   SFÎRŞIT
RÎNDLIN         END
EDIT-COL        SEQ
C7              PENTRU I DE LA 1 LA 3
                     Q=5 * (I+1)
                     PRINT AT (17, Q); D(I)
C7              SFÎRŞIT
EDIT-COL        END
MAX-LIV         PENTRU Z DE LA 1 LA 6
CALC-MAX             PENTRU I DE LA 1 LA 3
MAX2                 DACĂ A (Z, I) > H
MAX3                    SEQ
                        H=A (Z, I)
                        R=Z
                        C=I
MAX3                    END
MAX2                 SFÎRŞIT
CALC-MAX        SFÎRŞIT
MAX-LIV         SFÎRŞIT
EDIT-MAX        SEQ
                PRINT AT (20, 1); "MAXIMUL LIVRĂRILOR"; H; "TONE"
                PRINT AT (22, 4); "REZERVOR=R"; C
                PRINT AT (24, 4); "ZIUA="; C$ (R)
EDIT-MAX        END
EX 6 PC AMIC    END
```

Fig. 6.2(h) continuare.

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (PRAE)

```
EX 6 PC PRAE      SEQ
   (v. EXEMPLUL 6 – PC (aMIC))
LIVRĂRI           PENTRU Z DE LA 1 LA 6
                     PRINT C$ (Z)
C                    PENTRU I DE LA 1 LA 3
C1                   SEQ
                     PRINT "R";
                     PRINT USING "#"; I;
                     PRINT " ";
                     INPUT A (Z, I)
C1                   END
C                    SFÎRŞIT
LIVRĂRI           SFÎRŞIT
   (v. EXEMPLUL 6 – PC (aMIC))
CAP-TAB           SEQ
                     PRINT "ZIUA R1 R2 R3 MEDIA"
C3                   PENTRU I DE LA 1 LA 28
                        PRINT " – ";
C3                   SFÎRŞIT
CAP-TAB           END
EDIT-RÎND         SEQ
C4                PENTRU Z DE LA 1 LA 6
                     PRINT C$ (Z)
C5                   PENTRU I DE LA 1 LA 3
                     PRINT TAB (5*I+5); A (Z, I)
C5                   SFÎRŞIT
                     PRINT TAB (23); B(Z)
C4                SFÎRŞIT
EDIT-RÎND         END
RÎNDLIN           SEQ
C6                PENTRU I DE LA 1 LA 28
                     PRINT " – ";
C6                SFÎRŞIT
RÎNDLIN           END
EDIT-COL          SEQ
C7                PENTRU I DE LA 1 LA 3
                     PRINT TAB (5*I+5); D(I);
C7                SFÎRŞIT
EDIT-COL          END
   (v. EXEMPLUL 6 – PC (aMIC))
EDIT-MAX          SEQ
                  PRINT "MAXIMUL LIVRĂRILOR"; H; "TONE"
                  PRINT "REZERVOR=R"; C
                  PRINT "ZIUA="; C$ (R)
EDIT-MAX          END
EX 6 PC PRAE      END
```

Fig. 6.2(i). *i)* pseudocodul programului BASIC-PRAE;

```
                              PSEUDOCODUL
          Nume program: EXEMPLUL 6 – PC (HC-85, TIM S, SPECTRUM)

EX 6 PC HC 85      SEQ
  (v. EXEMPLUL 6 – PC (aMIC)
LIVRĂRI            PENTRU Z DE LA 1 LA 6
                   PRINT C$(Z); " : " ;
C                  PENTRU I DE LA 1 LA 3
CC                 SEQ
                   PRINT "A ("; Z; ", " ; ")=" ;
                   INPUT A (Z, I)
CC                 END
C                  SFÎRȘIT
LIVRĂRI            SFÎRȘIT
                   (v. EXEMPLUL 6 – PC (aMIC)
EDIT–MAX           SEQ
                   PRINT AT (19, 1); "MAXIMUL LIVRĂRILOR="; H; "TONE"
                   PRINT AT (20, 4); "REZERVOR=R"; C
                   PRINT AT (21, 4); "ZIUA=" C$ (R)
EDIT–MAX           END
EX 6 PC HC 85      END
```

j)

```
                              PSEUDOCODUL
          Nume program: EXEMPLELE 6 – PC (AMSTRAD), m, F

EXEMPLUL6          SEQ
CIT–ZILE           PENTRU I DE LA 1 LA 6
                      READ C$ (I)
CIT–ZILE           SFÎRȘIT
LIVRĂRI            PENTRU Z DE LA 1 LA 6
                      S=0
C                     PENTRU I DE LA 1 LA 3
                         INPUT A (Z, I)
                         S=S+A(Z, I)
C                     SFÎRȘIT
                      B(Z)=S/3
LIVRĂRI            SFÎRȘIT
PREL–REZ           PENTRU I DE LA 1 LA 3
                      S=0
E                     PENTRU Z DE LA 1 LA 6
                         S=S+A (Z, I)
E                     SFÎRȘIT
                      D(I)=S/6
PREL–REZ           SFÎRȘIT
EDIT–RIND          PENTRU Z DE LA 1 LA 6
                      PRINT C$(Z), A(Z,1), A(Z,2), A(Z,3), B(Z)
EDIT–RIND          SFÎRȘIT
G                  PENTRU M DE LA 1 LA 70
                      PRINT " – " ;
G                  SFÎRȘIT
EDIT–COL           PENTRU I DE LA 1 LA 3
                      PRINT D(I)
EDIT–COL           SFÎRȘIT
```

k)

Fig. 6.2. *j)* pseudocodul programului BASIC HC-85, TIM S, SPECTRUM; *k)* pseudocodul programului BASIC-AMSTRAD, BASIC-80, ABASIC.

PSEUDOCODUL

Nume program: EXEMPLELE 6 – PC (AMSTRAD), m, F

```
K               SEQ
                  H=A(1,1)
                  R=1
                  C=1
K               END
MAX–LIV         PENTRU Z DE LA 1 LA 6
CALC–MAX          PENTRU I DE LA 1 LA 3
MAX 2               DACĂ A (Z, I) > H
MAX 3               SEQ
                      H=A(Z,I)
                      R=Z
                      C=I
MAX 3               END
MAX 2               SFÎRȘIT
CALC–MAX          SFÎRȘIT
MAX–LIV         SFÎRȘIT
                Afișează cap-tabel
EDIT–MAX        SEQ
                  PRINT H
                  PRINT C
                  PRINT C$(R)
EDIT–MAX        END
EXEMPLUL 6      END
```

k) continuare

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (COMMODORE)

```
EX 6 PC COMMO   SEQ
  (v. EXEMPLUL 6 – PC (PRAE)
LIVRĂRI         PENTRU Z DE LA 1 LA 6
                  PRINT C$(Z); " : "
C               PENTRU I DE LA 1 LA 3
CC              SEQ
                PRINT "A (" ; Z ; " , " ; ")=";
                INPUT A (Z, I)
CC              END
C               SFÎRȘIT
LIVRĂRI         SFÎRȘIT
  (v. EXEMPLUL 6 – PC (PRAE))
EX 6 PC COMMO   END
```

l)

Fig. 6.2(l). *l)* pseudocodul programului BASIC-COMMODORE;

PSEUDOCODUL

Nume program: EXEMPLUL 6 – M

```
EX 6 PLUS   SEQ
   (v. EXEMPLUL 6 – PC (AMSTRAD), m, F)
LIVRĂRI     PENTRU Z DE LA 1 LA 6
L1          SEQ
              PRINT C$(Z)
              S=0
              INPUT "REZERVORUL 1"; A(Z,1)
              S=S+A(Z,1)
              INPUT "REZERVORUL 2"; A(Z,2)
              S=S+A(Z,2)
              INPUT "REZERVORUL 3"; A(Z,3)
              S=S+A(Z,3)
              PRINT
              B(Z)=S/3
L1          END
LIVRĂRI     SFIRŞIT
   (v. EXEMPLUL 6 – PC (AMSTRAD), m, F)
EX 6 PLUS   END
```

Fig. 6.2. *m)* pseudocodul programului BASIC-PLUS.

□ Codificarea în limbajul BASIC–aMIC

vol. 2, pag. 218

Variabile indexate alfanumerice (şir)

Variabilele şir (alfanumerice) vă permit să manipulaţi date alfanumerice. Le puteţi utiliza într-una din următoarele instrucţiuni BASIC: **DIM, LET, PRINT, INPUT, READ, DATA, IF–THEN–ELSE** etc. Pentru calculatorul aMIC numele unei variabile şir este format dintr-o literă urmată de caracterul " **$** ".

Puteţi utiliza şi tablouri de variabile şir, tablourile fiind formate din mai multe variabile şir, cu aceeaşi lungime şi acelaşi nume. Trebuie avut grijă ca orice tablou de variabile şir pe care-l folosim să-l declarăm în instrucţiunea **DIM,** precizînd atît numărul de variabile cît şi lungimea acestora. Aşa se şi explică prezenţa instrucţiunii

```
5 DIM C$(6, 8), B(6), D(3)
```

în cadrul programului în care declarăm vectorul **C$** cu şase elemente (zilele lucrătoare ale săptămînii) de lungime 8 caractere (v. ziua de miercuri – cea mai lungă!)

Observaţie. Tablourile numerice şi cele şir pot fi declarate intercalat, în aceeaşi instrucţiune **DIM** (v. linia 5).

Puteţi introduce valori pentru variabilele şir (indexate) folosind una din instrucţiunile **INPUT** sau **READ.**

Cu secvenţa de instrucţiuni:

```
 9 FOR I=1 TO 6
10   READ C$(I)
12 NEXT I
15 DATA LUNI, MARŢI, MIERCURI,
   JOI, VINERI, SÎMBĂTĂ
```

se citesc (static) în variabilele indexate C$(1), C$(2), . . . , C$(6) valorile: LUNI, MARŢI, . . . , SÎMBĂTĂ.

Reguli

- Variabilele şir (exemplu A$) nu trebuie declarate într-o instrucţiune **DIM,** ele putînd fi utilizate direct în instrucţiunile de atribuire sau de intrare/ieşire;
- Tablourile şir trebuie declarate într-o instrucţiune **DIM** înainte de utilizare (folosirea lor se face prin specificarea unei anumite variabile indexate şir şi nu a întregului tablou);
- La execuţia instrucţiunii **DIM** pentru tablouri de variabile şir se alocă cîte un octet pentru fiecare caracter (element);
- Dacă expresia şir atribuită unei variabile şir, este mai lungă decît dimensiunea variabilei (stabilită prin **DIM** sau printr-o atribuire anterioară), atunci se atribuie variabilei şir doar prima parte din expresia şir (egală cu lungimea variabilei) restul expresiei şir ignorîndu-se;
- Dacă lungimea şirului din membrul drept este mai mică decît lungimea variabilei şir, restul de caractere se completează cu spaţii;
- Este posibilă utilizarea unor subşiruri (porţiuni) dintr-un şir desemnat de o variabilă şir. Se foloseşte notaţia

⟨n1⟩ **TO** ⟨n2⟩

unde:
⟨n1⟩ reprezintă indicele primului caracter al subşirului;
⟨n2⟩ este indicele ultimului caracter al subşirului.

Observaţie. Indicii n1 şi n2 pot fi constante, variabile sau expresii, valoarea lor fiind cuprinsă între 1 şi lungimea şirului. În cazul absenţei indicelui ⟨n1⟩ se consideră că subşirul începe cu primul caracter al şirului iar în cazul absenţei indicelui ⟨n2⟩ se consideră că subşirul se termină cu ultimul carater al subşirului. Se pot utiliza subşiruri ale variabilelor şir ce formează un tablou de variabile şir. Pentru subşiruri formate dintr-un singur caracter, se poate folosi un singur indice, cel al caracterului selectat.

PRINT AT

Cu instrucţiunea **PRINT AT** puteţi tipări oriunde pe suprafaţa ecranului (32 linii×30 coloane) valori, mesaje explicative etc. indiferent de poziţia curentă, în care urmau să se afişeze acestea.

Formatul general al instrucţiunii este:

Format general
PRINT AT (⟨nr. linie⟩, ⟨nr. coloană⟩)

în care:
⟨nr. linie⟩ reprezintă numărul de linie, cu valori cuprinse între 1 şi 32;
⟨nr. coloană⟩ reprezintă numărul de coloană cu valori cuprinse între 1 şi 30.

Observaţie. Punctul de coordonate (1,1) se află în colţul din stînga sus al ecranului iar punctul de coordonate (32, 30) în dreapta jos.

În program instrucţiunea **PRINT AT** s-a folosit în liniile 85–95; 405; 415; 510; 525; 535; 605; 715; 855–865.

Aplicaţie. Analizaţi singuri, fără ajutorul nostru, instrucţiunile 85–95, 405, 415 din cadrul programului şi simulaţi cu creionul şi hîrtia modul cum funcţionează fiecare dintre ele.

Reguli

- Într-o instrucţiune **PRINT AT** (limbajul BASIC-aMIC) coordonatele pentru ⟨nr. linie⟩ şi ⟨nr. coloană⟩ se transmit între paranteze;
- **PRINT AT** este foarte des întrebuinţată în afişarea de valori sau texte (mesaje) pe un desen realizat cu instrucţiunile grafice BASIC-aMIC (v. conversaţiile 12, 13).

Instrucţiuni matriciale. MAT INPUT

Pentru citirea dinamică a matricei A (6×3 elemente) vom utiliza instrucţiunea matricială **MAT INPUT**.

100 **MAT INPUT** A (6, 3)

De notat că matricea A nefiind declarată în instrucţiunea **DIM** din linia 5 a trebuit definită (specificate dimensiunile) în instrucţiunea **MAT** (v. linia 100).

În Fig. 6.3 se prezintă o imagine a elementelor componente ale acestui tablou ca rezultat al execuţiei instrucţiunii **MAT INPUT** din program.

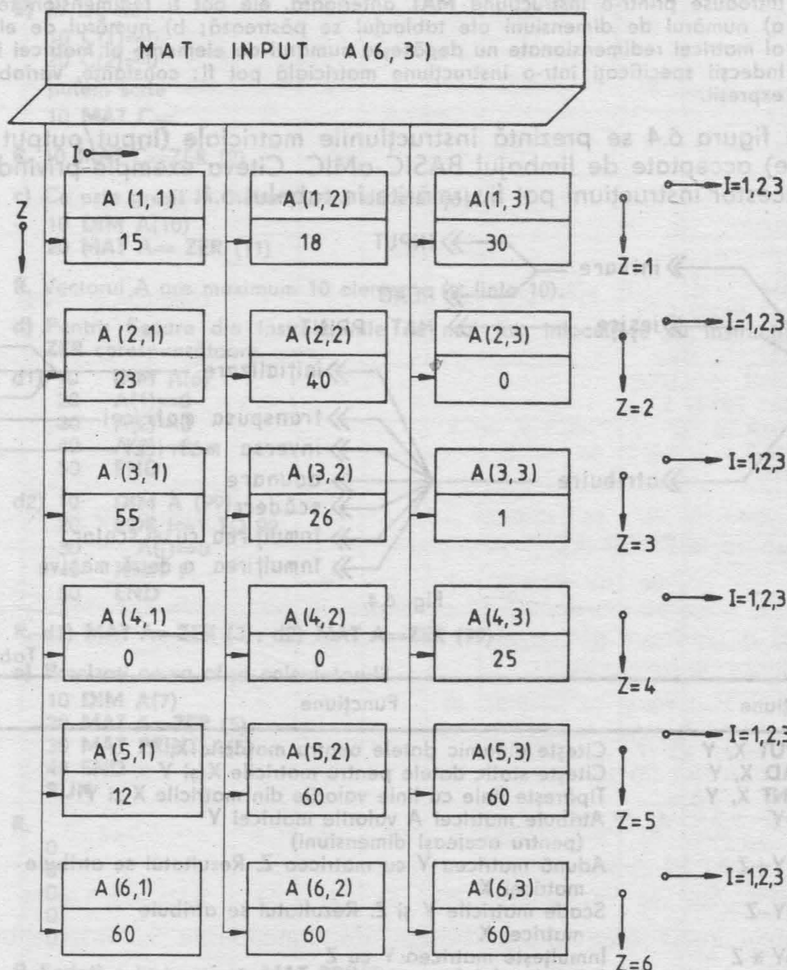


Fig. 6.3.

Observaţie. Instrucţiunea **MAT INPUT** spre deosebire de instrucţiunile cunoscute pentru prelucrarea tablourilor nu face referiri la fiecare element al tabloului.

În BASIC-aMIC instrucţiunile matriciale pot fi împărţite în două categorii: a) instrucţiuni de intrare/ieşire (**MAT INPUT, MAT READ, MAT PRINT**); b) instrucţiuni de atribuire – în care matricei din membrul stîng i se atribuie rezultatul unei operaţii matriciale (Exemplu: **MAT** X=Y; **MAT** X=Y * Z etc.).

Reguli

- Datorită prefixului **MAT,** utilizat în instrucţiuni, tablourile vor fi numite matrice (indiferent dacă au una sau două dimensiuni);
- Matricele introduse prin instrucţiunile de intrare/ieşire sau matricea din membrul stîng al unei instrucţiuni de atribuire, pot fi alocate la execuţia instrucţiunii **MAT** respective, nefiind necesară declararea lor în instrucţiunea **DIM**;
- În cazul în care matricile au fost declarate în instrucţiunea **DIM,** sau au fost introduse printr-o instrucţiune **MAT** anterioară, ele pot fi redimensionate dacă: a) numărul de dimensiuni ale tabloului se păstrează; b) numărul de elemente al matricei redimensionate nu depăşeşte numărul de elemente al matricei iniţiale;
- Indecşii specificaţi într-o instrucţiune matricială pot fi: constante, variabile sau expresii.

În figura 6.4 se prezintă instrucţiunile matriciale (input/output şi de atribuire) acceptate de limbajul BASIC-aMIC. Cîteva exemple privind utilizarea acestor instrucţiuni pot fi urmărite în tabelul 6.1.

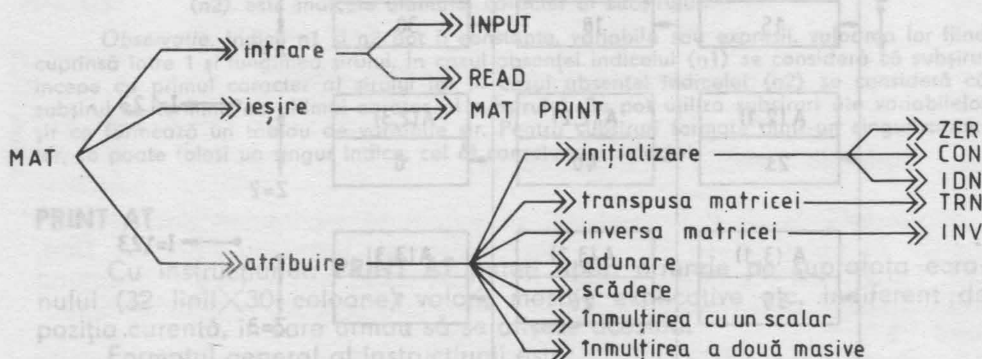


Fig. 6.4.

Tabelul 6.1

Instrucţiune	Funcţiune
MAT INPUT X, Y	Citeşte dinamic datele pentru matricile X, Y
MAT READ X, Y	Citeşte static datele pentru matricile X şi Y
MAT PRINT X, Y	Tipăreşte linie cu linie valorile din matricile X şi Y
MAT X=Y	Atribuie matricei A valorile matricei Y (pentru aceleaşi dimensiuni)
MAT X=Y+Z	Adună matricea Y cu matricea Z. Rezultatul se atribuie matricei X
MAT X=Y–Z	Scade matricile Y şi Z. Rezultatul se atribuie matricei X
MAT X=Y * Z	Înmulţeşte matricea Y cu Z
MAT X=⟨expresie⟩ * Y	Înmulţeşte matricea Y cu scalarul ⟨expresie⟩
MAT X=**INV**(B)	Inversează matricea pătrată B
MAT X=**TRN**(Y)	Calculează transpusa matricei Y
MAT X=**ZER**	Iniţializează cu zero elementele matricei X
MAT X=**CON**	Iniţializează cu unu elementele matricei X
MAT X=**IDN**	Iniţializează cu unu elementele diagonalei principale

TEST

a) În programul

```
10  DIM C(20)
20  READ N
30  FOR I=1 TO N
40     C(I)=0
50  NEXT I
60  DATA 7
70  END
```

liniile 30, 40, 50 pot fi înlocuite cu o singură instrucţiune **MAT**;
30 **MAT** C=

R. 30 **MAT** C=**ZER** (N)

b) În loc de:

```
10 C(1)=0
20 C(2)=0
```

putem scrie
10 **MAT** C=

R. 10 **MAT** C=**ZER** (2)

c) Ce este greşit în instrucţiunea de mai jos?

```
10 DIM A(10)
20 MAT A= ZER (11)
```

R. Vectorul A are maximum 10 elemente (v. linia 10).

d) Pentru fiecare din instrucţiunile de mai jos, înlocuiţi-le cu instrucţiunea **MAT ZER** corespunzătoare.

d1)
```
10  DIM A(6)
20  A(1)=0
30  A(2)=0
40  A(3)=0
50  END
```

d2)
```
10  DIM A (99)
20  FOR I=1 TO 99
30     A(I)=0
40  NEXT I
50  END
```

R. d1) **MAT** A=**ZER** (3); d2) **MAT** A=**ZER** (99)

e) Precizaţi ce va afişa calculatorul?

```
10 DIM A(7)
20 MAT A=ZER (5)
30 MAT PRINT A(5)
40 END
RUN
```

R.

```
0
0
0
0
0
```

f) Scrieţi o instrucţiune **MAT PRINT** care să înlocuiască următoarea buclă **FOR-NEXT** (se omit etichetele).

```
FOR I=1 TO N
    PRINT C (I)
NEXT I
```

R. MAT PRINT C(N)

g) Scrieţi o instrucţiune **MAT INPUT** care să înlocuiască următoarea buclă **FOR-NEXT.**

```
FOR I=1 TO M
  INPUT A(I)
NEXT I
```

R. MAT INPUT A (M)

Aplicaţie. Să se rezolve sistemul de ecuaţii

$$\begin{cases} 2x-9y-5z=2 \\ 7x-6y+5z=-35 \\ 9x-6y+5z=-39. \end{cases}$$

Fie C matricea coeficienţilor, S vectorul (coloană) soluţiei şi K vectorul termenilor liberi.

$$C=\begin{bmatrix} 2 & -9 & -5 \\ 7 & -6 & +5 \\ 9 & -6 & +5 \end{bmatrix} \quad S=\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad K=\begin{bmatrix} 2 \\ -35 \\ -39 \end{bmatrix}$$

Sistemul devine:

$$C*S=K \qquad (1)$$

sau

$$C^{-1}*C*S=C^{-1}*K \qquad (2)$$

sau

$$I*S=C^{-1}*K \qquad (3)$$

de unde rezultă:

$$S=C^{-1}*K \qquad (4)$$

```
10 DIM C(3,3), S(3,1), K(3,1), N(3,3)
20 MAT READ C, K
30 MAT N= INV (C)
40 MAT S= N*K
50 PRINT "SOLUŢIE"
60 MAT PRINT S
70 DATA 2, -9, -5, 7, -6, 5, 9, -6, 5
80 DATA 2, -35, -39
90 END
```

Revenind la PRINT AT

Pentru afişarea pe suprafaţa ecranului în puncte de coordonate date (calculate) a livrărilor zilnice şi a livrărilor medii calculate, pe zile, s-au scris următoarele instrucţiuni:

```
500 FOR Z=1 TO 6
505   P=8+Z
510   PRINT AT (P; 1); C$(Z)
515   FOR I=1 TO 3
520     Q=5*I
525     PRINT AT (P, Q); A(Z, I)
530   NEXT I
535   PRINT AT (P, 26); B (Z)
540 NEXT Z
```

Pentru înţelegerea mecanismului de funcţionare a instrucţiunilor **PRINT AT:**

```
510 PRINT AT (P, 1); C$ (Z)
525 PRINT AT (P, Q); A(Z,I)
535 PRINT AT (P,26); B(Z)
```

consultaţi fig. 6.5.

NR. LINIE	INSTRUCTIUNE	Z	P	I	Q	C$(Z)	B(Z)	A(Z,I)	
505	P = 8+Z	1	9	–	–	–	–	–	
510	PRINT AT(P,1); C$(Z)	1	9	–	–	LUNI	–		1 — 9 LUNI
520	Q = 5 * I	1	9	1	5	LUNI	–		
510	PRINT AT(P,Q); A(Z,I)	1	9	1	5	LUNI	–	15	1 5 — 9 LUNI 15
520	Q = 5 * I	1	9	2	10	LUNI	–	15	
510	PRINT AT(P,Q); A(Z,I)	1	9	2	10	LUNI	–	18	1 5 10 — 9 LUNI 15 18
520	Q = 5 * I	1	9	3	15	LUNI	–	18	
510	PRINT AT(P,Q); A(Z,I)	1	9	3	15	LUNI	–	30	1 5 10 15 — 9 LUNI 15 18 30
535	PRINT AT(P,26); B(Z)	1	9	4	15	LUNI	21	30	1 5 10 15 26 — 9 LUNI 15 18 30 21
⋮									
505	P = 8+Z	6	14	—	—	VINERI	40	60	
510	PRINT AT(P, 1); C$(Z)	6	14	—	—	SIMBATA	40	60	1 — 14 SIMBATĂ
520	Q = 5 * I	6	14	1	5	SIMBATA	40	60	
510	PRINT AT(P,Q); A(Z,I)	6	14	1	5	SIMBATA	40	60	1 5 — 14 SIMBATĂ 60
520	Q = 5 * I	6	14	2	10	SIMBATA	40	60	
510	PRINT AT(P,Q); A(Z,I)	6	14	2	10	SIMBATA	40	60	1 5 10 — 14 SIMBATĂ 60 60
520	Q = 5 * I	6	14	3	15	SIMBATA	40	60	
510	PRINT AT(P,Q); A(Z,I)	6	14	3	15	SIMBATA	40	60	1 5 10 15 — 14 SIMBATĂ 60 60 60
535	PRINT AT(P,26); B(Z)	6	14	4	15	SIMBATA	60	60	1 5 10 15 26 — 14 SIMBATĂ 60 60 60 60

Fig. 6.5.

Cu setul de instrucţiuni:

```
600 FOR M=1 TO 30
605    PRINT AT (16, M); "–"
610 NEXT M
```

se trasează pe linia 16 a ecranului un rînd de "–", începînd cu coloana 1 pînă în coloana 30 inclusiv. Pentru afişarea livrărilor medii calculate, (pe rezervor) s-au scris instrucţiunile:

```
700 PRINT AT (17, 3); "MEDIA"
705 FOR I=1 TO 3
710    Q=5*(I+1)
715    PRINT AT (17, Q); D (I)
720 NEXT I
```

În figura 6.6 se ilustrează modul în care poate fi utilizată instrucţiunea **PRINT AT**

```
715 PRINT AT (17, Q): D(I)
```

NR.LINIE	INSTRUCTIUNE	I	Q	D(I)	
710	Q = 5 * (I + 1)	1	10	27.5	10
715	PRINT AT(17,Q); D(I)	1	10	27.5	17 27.5
710	Q = 5 * (I + 1)	2	15	34	10 15
715	PRINT AT(17,Q), D(I)	2	15	34	17 27.5 34
710	Q = 5 * (I + 1)	3	20	29.3333	10 15 20
715	PRINT AT(17,Q); D(I)	3	20	29.3333	17 27.5 34 29.3333

Fig. 6.6.

atunci cînd unul din parametri (17) se menţine constant.

Aplicaţie

Completaţi programul cu următoarea secvenţă de instrucţiuni:

```
18 INIT
20 FOR J=1 TO 4
25    PRINT AT (15, 2); "    ?    "
30    PRINT AT (16, 2); "    ?    "
35    PRINT AT (17, 2); "    ?    "
40    PRINT AT (18, 2); "    ?    "
45    FOR K=1 TO 150
50    NEXT K
55    INIT
60 NEXT J
```

Introduceţi în instrucţiunile 25, 30, 35, 40 (v. semnul întrebării) – caracterele semigrafice (de pe tastatură) corespunzătoare afişării mesajului:

EXEMPLUL 6

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 DIM X(3,4), Y(6,10), Z(8,9)
20 READ L, C
30 MAT READ X, Y(2,5), Z(L,C)
40 MAT PRINT Y, Y; Z
50 DATA 4,3
60 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
70 DATA 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
80 DATA 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34
90 END
```

b)
```
10 FOR I=1 TO 4
20   FOR J=I TO 4
30     A (I, J)=-8
40   NEXT J
50 NEXT I
60 FOR I=1 TO 4
70   FOR J=1 TO 4
80     PRINT A (I, J);
90   NEXT J
95   PRINT
96   PRINT
100 NEXT I
110 END
```

c)
```
10 DIM A (4,4)
20 MAT READ A
30 MAT PRINT A;
40 DATA -8, -8, -8, -8, -8, -8, -8, -8
50 DATA -8, -8, -8, -8, -8, -8, -8, -8
60 END
```

Prelucrarea matricelor. Determinarea maximului livrărilor

În cadrul programului intervin prelucrări ale datelor organizate în vectori şi matrici.

Astfel, valorile livrărilor medii pe zile calculate în bucla **FOR-NEXT** din liniile 200–230 se depun în variabila B(Z),

```
200 FOR Z=1 TO 6
205     S=0
210     FOR I=1 TO 3
215       S=S+A(Z,I)
220     NEXT I
225     B(Z)=S/3
230 NEXT Z
```

iar valorile livrărilor medii pe fiecare rezervor se depun în variabila D(I).

```
300 FOR I=1 TO 3
305   S=0
310   FOR Z=1 TO 6
315     S=S+A(Z,I)
320   NEXT Z
325   D(I)=S/6
330 NEXT I
```

În tabelele 6.2, 6.3 şi figurile 6.7, 6.8 puteţi urmări modul de desfăşurare a calculelor pentru fiecare caz în parte.

Tabelul 6.2

Nr. linie	Instrucțiune BASIC	Z	I	A	S	B
0	1	2	3	4	5	6
205	S=0	1	0	0	0	0
215	S=S+A(Z,I)	1	1	A(1,1) 15	A(1,1) 15	0
215	S=S+A(Z,I)	1	2	A(1,2) 18	A(1,1)+A(1,2) 15+18	0
215	S=S+A(Z,I)	1	3	A(1,3) 30	A(1,1)+A(1,2)+A(1,3) 15+18+30	0
225	B(Z)=S/3	1	4	A(1,3) 30	A(1,1)+A(1,2)+A(1,3) 15+18+30	21
205	S=0	2	4	–	0	21
215	S=S+A(Z,I)	2	1	A(2,1) 23	A(2,1) 23	21
215	S=S+A(Z,I)	2	2	A(2,2) 40	A(2,1)+A(2,2) 23+40	21
215	S=S+A(Z,I)	2	3	A(2,3) 0	A(2,1)+A(2,2)+A(2,3) 23+40+0	21
225	B(Z)=S/3	2	4	A(2,3) 0	A(2,1)+A(2,2)+A(2,3) 23+40+0	21
205	S=0	3	4	–	–	–
215	S=S+A(Z,I)	3	1	A(3,1) 55	A(3,1) 55	21
215	S=S+A(Z,I)	3	2	A(3,2) 26	A(3,1)+(3,2) 55+26	21
215	S=S+A(Z,I)	3	3	A(3,3) 1	A(3,1)+A(3,2)+A(3,3) 55+26+1	21
225	B(Z)=S/3		4	–	A(3,1)+A(3,2)+A(3,3) 55+26+1	27.333

Tabelul 6.2 (continuare)

0		2	3	4	5	6
205	S=0	4	4	–	0	27.3333
215	S=S+A(Z,I)	4	1	A(4,1) 0	A(4,1) 0	27.3333
215	S=S+A(Z,I)	4	2	A(4,2) 0	A(4,1)+A(4,2) 0+0	27.3333
215	S=S+A(Z,I)	4	3	A(4,3) 25	A(4,1)+A(4,2)+A(4,3) 0+0+25	27.3333
225	B(Z)=S/3	4	4	–	A(4,1)+A(4,2)+A(4,3) 0+0+25	8.3333
205	S=0	5	4	–	0	8.3333
215	S=S+A(Z,I)	5	1	A(5,1) 12	A(5,1) 12	8.3333
215	S=S+A(Z,I)	5	2	A(5,2) 60	A(5,1)+A(5,2) 12+60	8.3333
215	S=S+A(Z,I)	5	3	A(5,3) 60	A(5,1)+A(5,2)+A(5,3) 12+60+60	8.3333
225	B(Z)=S/3	5	4	–	A(5,1)+A(5,2)+A(5,3) 12+60+60	44
205	S=0	6	4	–	0	44
215	S=S+A(Z,I)	6	1	A(6,1) 60	A(6,1) 60	44
215	S=S+A(Z,I)	6	2	A(6,2) 60	A(6,1)+A(6,2) 60+60	44
215	S=S+A(Z,I)	6	3	A(6,3) 60	A(6,1)+A(6,2)+A(6,3) 60+60+60	44
225	B(Z)=S/3	6	4	–	A(6,1)+A(6,2)+A(6,3) 60+60+60	60

Tabelul 6.3

Nr. linie	Instrucțiune BASIC	I	Z	A	S	D
0	1	2	3	4	5	6
305	S=0	1	0	0	0	0
315	S=S+A(Z,I)	1	1	A(1,1) 15	A(1,1) 15	0
315	S=S+A(Z,I)	1	2	A(2,1) 23	A(1,1)+A(2,1) 15+23	0
315	S=S+A(Z,I)	1	3	A(3,1) 55	A(1,1)+A(2,1)+A(3,1) 15+23+55	0
315	S=S+A(Z,I)	1	4	A(4,1) 0	A(1,1)+A(2,1)+A(3,1)+A(4,1) 15+23+55+0	0
315	S=S+A(Z,I)	1	5	A(5,1) 12	A(1,1)+A(2,1)+A(3,1)+A(4,1)+A(5,1) 15+23+55+0+12	0
315	S=S+A(Z,I)	1	6	A(6,1) 60	A(1,1)+A(2,1)+A(3,1)+A(4,1)+A(5,1)+A(6,1) 15+23+55+0+12+60	0
325	D(I)=S/6	1	7	–	15+23+55+0+12+60	27.5
305	S=0	2	7	–	0	27.5
315	S=S+A(Z,I)	2	1	A(1,2) 18	A(1,2) 18	27.5
315	S=S+A(Z,I)	2	2	A(2,2) 40	A(1,2)+A(2,2) 18+40	27.5
315	S=S+A(Z,I)	2	3	A(3,2) 26	A(1,2)+A(2,2)+A(3,2) 18+40+26	27.5

315	S=S+A(Z,I)	2	4	A(4,2) 0	A(1,2)+A(2,2)+A(3,2)+A(4,2) 18+40+26+0	27.5
315	S=S+A(Z,I)	2	5	A(5,2) 60	A(1,2)+A(2,2)+A(3,2)+A(4,2)+A(5,2) 18+40+26+0+60	27.5
315	S=S+A(Z,I)	2	6	A(6,2) 60	A(1,2)+A(2,2)+A(3,2)+A(4,2)+A(5,2)+A(6,2) 18+40+26+0+60+60	
325	D(I)=S/6					34
305	S=0	3	7		0	34
315	S=S+A(Z,I)	3	1	A(1,3) 30	A(1,3) 30	34
315	S=S+A(Z,I)	3	2	A(2,3) 0	A(1,3)+A(2,3) 30+0	34
315	S=S+A(Z,I)	3	3	A(3,3) 1	A(1,3)+A(2,3)+A(3,3) 30+0+1	34
315	S=S+A(Z,I)	3	4	A(4,3) 25	A(1,3)+A(2,3)+A(3,3)+A(4,3) 30+0+1+25	34
315	S=S+A(Z,I)	3	5	A(5,3) 60	A(1,3)+A(2,3)+A(3,3)+A(4,3)+A(5,3) 30+0+1+25+60	34
315	S=S+A(Z,I)	3	6	A(6,3) 60	A(1,3)+A(2,3)+A(3,3)+A(4,3)+A(5,3)+A(6,3) 30+0+1+25+60+60	34
325	D(I)=S/6					29.333

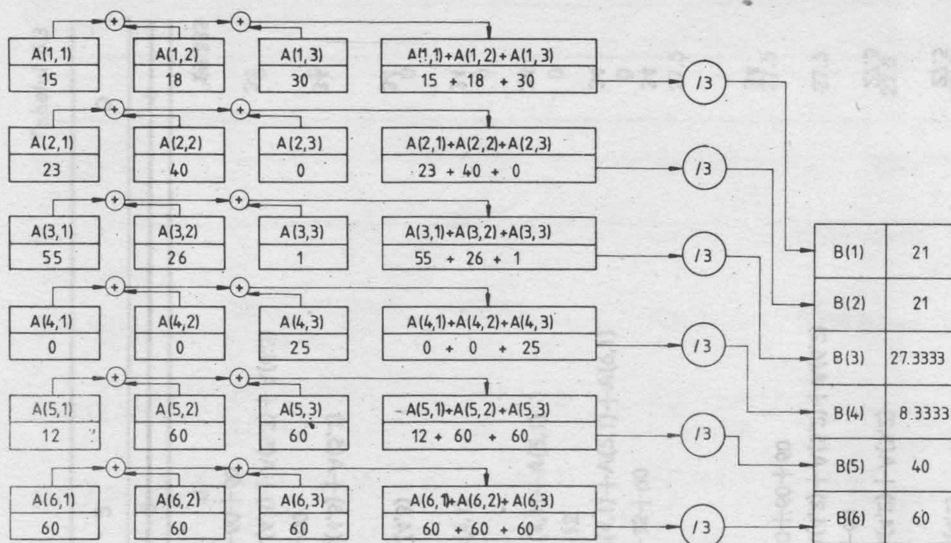


Fig. 6.7.

A(1,1)	A(1,2)	A(1,3)
15	18	30
A(2,1)	A(2,2)	A(2,3)
23	40	0
A(3,1)	A(3,2)	A(3,3)
55	26	1
A(4,1)	A(4,2)	A(4,3)
0	0	25
A(5,1)	A(5,2)	A(5,3)
12	60	60
A(6,1)	A(6,2)	A(6,3)
60	60	60
A(1,1)+A(2,1)+A(3,1)+A(4,1)+A(5,1)+A(6,1)	A(1,2)+A(2,2)+A(3,2)+A(4,2)+A(5,2)+A(6,2)	A(1,3)+A(2,3)+A(3,3)+A(4,3)+A(5,3)+A(6,3)
15 + 23 + 55 + 0 + 12 + 60	18 + 40 + 26 + 0 + 60 + 60	30 + 0 + 1 + 25 + 60 + 60
/6	/6	/6
D(1)	D(2)	D(3)
27.5	34	29.3333

Fig. 6.8.

Cu secvenţa de instrucţiuni:

```
800 H=A (1,1)
805 R=1
810 C=1
815 FOR Z=1 TO 6
820   FOR I=1 TO 3
825     IF A (Z, I)<=H THEN 845
830     H=A (Z, I)
835     R=Z
840     C=I
845   NEXT I
850 NEXT Z
```

se determină, în variabila H, maximul livrărilor. Valoarea finală a variabilelor R şi C indică numărul rezervorului respectiv ziua cînd s-a atins un maxim al livrărilor.

□ Codificarea în limbajul BASIC–PRAE

vol. 2, pag. 218

Variabile indexate şir (alfanumerice)

Numele unei variabile şir (alfanumerice) se defineşte la fel ca şi numele unei variabile numerice cu deosebirea că la o variabilă şir se pune după acest nume semnul dolarului. Variabilele şir ocupă loc imediat după variabilele numerice. Pe doi octeţi se scrie din nou numele variabilei, urmat nu de valoarea efectivă a acestei variabile ci de un aşa-zis „pointer" (arătător) care ne indică unde anume se găseşte această valoare. De notat că, valoarea unei variabile şir se poate găsi în program sau într-o zonă a memoriei numită zona de manevră pentru şiruri. Limbajul BASIC-PRAE permite definirea variabilelor indexate de tip şir pentru prelucrarea vectorilor şi masivelor de şiruri de caractere.

Reguli

- Variabila indexată de tip şir constă dintr-un nume de variabilă BASIC-PRAE, caracterul $ şi unul sau doi indici între paranteze;
- Limitele între care pot varia indicii unei variabile indexate de tip şir pot fi declarate: a) implicit (prin prima referinţă); b) explicit (prin instrucţiunea **DIM**);
- Expresiile de indici nu pot fi decît expresii aritmetice;
- Este posibilă utilizarea simultană a unei variabile simple şir şi a unei variable indexate şir cu acelaşi nume;
- Variabilele de tip şir sînt iniţializate la execuţia unui program cu valori egale cu "şirul vid" (şirul de caractere avînd lungimea zero);
- Variabilele şir nu trebuie declarate într-o instrucţiune **DIM**;
- Este posibilă utilizarea unor subşiruri dintr-un şir desemnat de o variabilă (v. limbajul BASIC-aMIC).

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 INPUT "NUME"; N$
20 FOR I=1 TO 3
```

```
30  READ A$, B$
35  IF N$=A$ THEN 70
40  NEXT I
50  PRINT "NUME NECUNOSCUT"
60  STOP
70  PRINT A$, B$
80  DATA ZAMFIRESCU, COLENTINA 12-BUCUREŞTI
90  DATA IONESCU, PARIS 13-BUCUREŞTI
100 DATA POPA, ROMA 10-BUCUREŞTI
120 END
```

PRINT AT

După cum am văzut în conversaţiile de pînă acum, instrucţiunea **PRINT** afişează întotdeauna rezultatele pe ecran începînd din ultimul rînd şi defilînd apoi ecranul. Instrucţiunea **PRINT AT** ne oferă posibilitatea de a alege punctul de pe ecran în care să se afişeze rezultate, mesaje etc.

Formatul general al instrucţiunii este

Format general
PRINT AT ⟨x⟩, ⟨y⟩; ⟨lista⟩

în care:

⟨x⟩, ⟨y⟩ reprezintă coordonatele punctului de pe ecran de unde dorim să scriem; ⟨lista⟩ are aceeaşi semnificaţie ca la **PRINT**.

Remarci

- Ecranul TV, din punctul de vedere al PRAE-ului este împărţit în 256 de puncte pe orizontală şi 256 de puncte pe verticală, deci un total de 256×256=65536 pixeli (pictural elements).
- Punctul de coordonate (0,0) este stabilit în colţul din stînga sus al ecranului.
- BASIC-PRAE acceptă ca limite pentru ⟨x⟩ şi ⟨y⟩ domeniul de valori 0÷255.
- Coordonatele x şi y nu se introduc între paranteze;
- ⟨x⟩ şi ⟨y⟩ pot fi expresii numerice.
- Deoarece pe ecran un caracter ocupă un cîmp de 8×8 puncte, rezultatelor expresiilor x şi y li se va considera partea întreagă, şi eventual vor fi micşorate pentru a obţine un număr multiplu de opt.

Înapoi la program

Diferenţele de scriere a programului apar în instrucţiunile de citire ale matricei de date A (limbajul BASIC-PRAE nu acceptă instrucţiuni matriciale!) cît şi în modul de utilizare a instrucţiunii **PRINT AT** (v. dimensiunile lui x şi y).

```
100 FOR Z=1 TO 6
110   PRINT, C$ (Z); " : "
120   FOR I=1 TO 3
130     PRINT "R";
135     PRINT USING "#"; I;
138     PRINT "   ";
140     INPUT A(Z, I)
150   NEXT I
160 NEXT Z
```

Observaţie. Remarcaţi în linia 135 utilizarea formatului de editare "#" pentru variabila I (indexul rezervorului). În limbajul BASIC-PRAE instrucţiunea **PRINT USING** admite specificaţii de format atît pentru numere cît şi pentru şiruri. Simbolurile folosite pentru editarea valorilor numerice sînt: "#", "–", ",", "*", ".", "∧". Ele au aceeaşi semnificaţie ca şi în limbajul BASIC–PLUS, BASIC-80.

În cadrul programului am preferat scrierii cu **PRINT AT** scrierea cu **PRINT TAB** în care am precizat doar coloana în care urmează să se afişeze rezultatele (v. liniile 515, 535, 710).

Remarcaţi de asemenea în linia 5 a programului dimensionarea matricei A (6×3 elemente).

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)
```
10 CLS
20 PRINT "COD"; "CANTITATE"; "PREŢ"
30 FOR I=1 TO 4
40    READ COD, CANTITATE, PREŢ
50    PRINT COD; CANTITATE; PREŢ
60 NEXT I
70 END
80 DATA 10, 3, 8
90 DATA 20, 4, 7
100 DATA 30, 6, 9
110 DATA 40, 5, 6
120 END
```

b) Modificaţi programul precedent de maniera:
```
20 PRINT "COD"; "CANTITATE"; "PREŢ"; "VALOARE"
50 PRINT COD; CANTITATE; PREŢ; CANTITATE*PREŢ
```
după care executaţi programul.

c)
```
10 PRINT "COD"; "DENUMIRE"; "CANTITATE"; "PREŢ"; "VALOARE"
20 FOR I=1 TO 4
30    READ COD, DENUMIRE$, CANTITATE, PREŢ
40    PRINT COD; DENUMIRE$; CANTITATE; PREŢ; CANTITATE * PREŢ
50 NEXT I
60 DATA 10, A1-B, 3, 82, 20, A2B, 4,7.50,30, A3-B, 6, 9.25, 40, A4-B, 5, 6.30
70 END
```

☐ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 219

Variabile indexate şir (alfanumerice)

În limbajul BASIC HC-85, TIM S, SPECTRUM variabilele şir şi variabilele indexate şir prezintă următoarele particularităţi:

- numele variabilelor şir şi al variabilelor indexate şir este format dintr-o literă urmată de semnul dolar;
- variabilele indexate şi variabilele şir pot avea lungimi şi număr de dimensiuni de lungime arbitrară;
- variabilele şir şi variabilele indexate şir nu pot avea acelaşi nume;
- rezervarea de spaţiu pentru o variabilă indexată şir se face prin utilizarea instrucţiunii **DIM**;
- variabilele şir sînt memorate pe un număr de:

3+(lungimea şirului)

octeţi;

- variabilele indexate şir sînt memorate pe un număr de 4+2*(numărul dimensiunilor)+(numărul total de caractere) octeţi;
- toate variabilele indexate şir vor avea aceeaşi lungime dată de ultima dimensiune **DIM**;
- la memorarera elementelor unui masiv (matrice) indicele dimensiunii cea mai din stînga variază cel mai încet;
- apelarea elementelor unui şir se face prin specificarea caracterului din şir, sau a poziţiei iniţiale şi finale a subşirului ce se apelează;
- este posibilă utilizarea unor subşiruri dintr-un şir desemnat de-o variabilă (v. limbajul BASIC-aMIC).

PRINT AT

În instrucţiunea **PRINT** din limbajul BASIC HC-85, TIM S, SPECTRUM se pot specifica linia (0–21) şi coloana (0–31) de unde dorim să se facă afişarea valorilor, folosind instrucţiunea **PRINT AT** (v. limbajul BASIC-PRAE).

Reguli

- BASIC HC-85, TIM S, SPECTRUM acceptă ca limite pentru x şi y domeniul de valori 0–21 (x), 0–31 (y).
- x şi y pot fi expresii numerice.
- Ecranul alfanumeric conţine 24 de linii a cîte 32 coloane (caractere).
- Liniile 22 şi 23 sînt rezervate mesajelor de sistem.
- Caracterul de adresă alfanumerică (0, 0) este fixat în colţul din stînga sus al ecranului.

Înapoi la program

În comparaţie cu programul BASIC-aMIC diferenţele de scriere apar în instrucţiunea **DATA** (constantele şir se pun între ghilimele).

```
15 DATA "LUNI", "MARŢI", "MIERCURI", "JOI", "VINERI", "SÎMBĂTĂ"
```

şi în instrucţiunile de citire a matricei de date A.

```
100 FOR Z=1 TO 6
110   PRINT C$ (Z); " : "
115   FOR I=1 TO 3
120     PRINT "A("; Z; ", "; I; ")=";
125     INPUT A(Z,I)
130   NEXT I
140   PRINT
150 NEXT Z
```

Observaţie. În BASIC HC-85, TIM S, SPECTRUM lipsesc instrucţiunile matriciale.

Deoarece ecranul alfanumeric conţine numai (0–21) linii operaţionale modificaţi liniile 855, 860, 865 ale programului BASIC-aMIC după cum urmează:

```
855 PRINT AT (19,1); "MAXIMUL LIVRĂRILOR="; H; "TONE"
860 PRINT AT (20,4); "REZERVOR=R"; C
865 PRINT AT (21,4); "ZIUA="; C$(R)
900 END
```

Aplicaţie. Introduceţi în program o secvenţă de instrucţiuni care să afişeze la începutul programului mesajul:

EXEMPLUL 6

scris cu caractere semigrafice (v. conversaţia 12).

Numere aleatoare

Numerele aleatoare reprezintă "inima" multor jocuri logice şi programe destinate, în special microcalculatoarelor şi calculatoarelor personale.

Dumneavoastră vă puteţi întîlni cu numerele aleatoare atunci cînd veţi juca jocuri ce implică aruncarea unui zar, aruncarea cu banul sau extragerea unor numere din pălărie.

Deşi toţi avem o idee despre ce înseamnă o succesiune de numere aleatoare este totuşi dificil să definim clar noţiunea.

Pentru a clarifica totuşi noţiunea vă propunem să aruncăm o privire asupra numerelor obţinute în urma aruncării unui zar (cu şase feţe) de 15 ori în două experimente "gîndite".

Să presupunem că în primul experiment au ieşit numerele:

1, 5, 2, 4, 6, 3, 2, 1, 6, 3, 5, 4, 3, 4, 2

iar în cel de-al doilea experiment s-au obţinut numerele:

6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6.

Dacă succesiunea de numere din primul experiment s-ar fi produs "la noroc" cît priveşte cel de-al doilea experiment începem să devenim suspicioşi şi să dezaprobăm zarul. Cu alte cuvinte, la sfîrşitul unei succesiuni mai lungi de numere ne aşteptăm să aplicăm "legile norocului"

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 FOR I=1 TO 5            RUN
20   LET X=RND             0. 85560608
30   PRINT X               0. 98418181
40 NEXT I                  0. 71151733
50 END                     0. 36412048
                           0. 66645813
```

În mod cert numerele de mai sus arată ca nişte numere aleatoare şi dacă veţi tasta **RUN** veţi obţine o listă complet diferită.

După cum constataţi, funcţia **RND** generează numere aleatoare în intervalul 0–1 cu 0 inclusiv dar cu 1 exclusiv.

Cum putem să mărim intervalul pentru a obţine şi alte numere aleatoare?

Foarte simplu, multiplicînd **RND** cu un număr.

Astfel,

RND generează un număr real aleator în intervalul 0,1;
5***RND** generează un număr real aleator în intervalul 0,6;
52***RND** generează un număr real aleator în intervalul 0,52.

Vă puteţi gîndi la **RND** ca la un "factor de conversie" care se schimbă după bunul său plac. Următorul program ilustrează această idee.

```
10 PRINT "i" ; TAB 8; "1*RND"; TAB 20; "6*RND"
20 PRINT "---"; TAB 8; "-----"; TAB 20; "-----"
30 FOR I=1 TO 5
40    LET A=RND
50    LET S=S*A
60    PRINT i; TAB 8; A; TAB 20; S
70 NEXT I
80 STOP
RUN

i      1*RND           6*RND
....   ...........     ...........
1      0.62315369      3.7389221
2      0.73695374      4.4217224
3      0.27182007      1.6309204
4      0.3873291       2.3239746
5      0.87016296      5.2209778
```

Executați programul înlocuind linia 50 cu:

```
50 LET S=52*A
RUN

i      1*RND           52*RND
....   ...........     ...........
1      0.65000916      33.800476
2      0.75108337      39.056335
3      0.33152771      17.239441
4      0.86534119      44.997742
5      0.90074158      46.838562
```

TEST

Scrieți un program care să afișeze șase numere aleatoare în intervalul 0÷5.999999.

```
10 FOR A=1 TO 6
20    LET X=6*RND
30    PRINT X
40 NEXT A
50 STOP
```

Aplicație. Introduceți și executați următoarele programe:

a)

```
10 DIM A(4)
20 FOR I=1 TO 4
30    LET A(I)=INT (RND*8)+1
40 NEXT I
50 FOR I=1 TO 4
60    PRINT TAB 6; "A("; I;")="
70 NEXT I
```

b)

```
10 DIM A(4,4)
20 FOR I=1 TO 4
30 FOR J=1 TO 4
40    LET A (I, J)=INT (RND * 8+1)
50    PRINT "A("; I; ","; J; ")="; A (I, J)
60 NEXT J
70 NEXT I
80 PRINT AT 16, 15; "1 2 3 4"
90 PRINT
100 FOR I=1 TO 4
110    PRINT TAB 13; I; TAB 15; A (I, 1); "   ";
         A (I, 2); "   "; A (I, 3); "   "; A (I, 4)
120 NEXT I
```

c)
```
10 DIM A (3, 3, 3)
20 FOR I=1 TO 3
30   FOR J=1 TO 3
40     FOR K=1 TO 3
50       LET A (I, J, K)=INT (RND*4)+1
60       PRINT "A("; I;","; J; ","; K; ")="; A (I, J, K)
70     NEXT K
80   NEXT J
90 NEXT I
100 STOP
```

d)
```
10 DIM A (2, 2, 2, 2, 2)
20 FOR I=1 TO 2
30 FOR J=1 TO 2
40 FOR K=1 TO 2
50 FOR L=1 TO 2
60 PRINT "A ("; I; ","; J; ","; K; ","; L; ","; M; ")="; A (I, J, K, L, M)
70 FOR M=1 TO 2
80 LET A (I,J, K, L, M)=INT (RND*9)+1
90 NEXT M
100 NEXT L
110 NEXT K
120 NEXT J
130 NEXT I
140 END
```

e)
```
10 DIM A$ (4, 10)
20 FOR I=1 TO 4
30   READ B$ (I)
40 NEXT I
50 FOR I=1 TO 4
60   PRINT "A$ ("; I; ")="; A$ (I)
70 NEXT I
80 DATA "ION", "PAUL", "AGA", "ADA"
90 STOP
```

f)
```
10 DIM B$ (21, 5)
20 FOR I=1 TO 21
30   READ A$(I)
40   IF 3*INT (I/3)=I THEN RESTORE
59 NEXT I
60 FOR I=21 TO 1 STEP-1
70   PRINT B$ (I); "  ";
80 NEXT I
90 DATA "PAUL", "DANI", "ADA"
```

g)
```
10 DATA "PI", 7
20 DIM B$ (21, 4)
30 DIM Z (21)
40 FOR I=1 TO 21
50   READ B$ (I), Z (I)
60   IF 3*INT (I/3)=I THEN RESTORE
65   DATA "DANI"
70 NEXT I
80 DATA 23, "DAN"
90 FOR I=1 TO 21
100   PRINT B$ (I), Z (I)
110   DATA 11
120 NEXT I
```

Observaţie. Instrucţiunea **RESTORE** (v. linia 60) face ca instrucţiunea **READ** următoare să citească datele de la o instrucţiune **DATA** aflată la linia ⟨n⟩ sau după aceasta. Dacă ⟨n⟩ lipseşte (cazul nostru) se ia ca valoare implicită 1.

Joc pe calculator

Scrieți un program care simulează aruncarea unui zar de 25 ori.

```
10 FOR I=1 TO 5
20   FOR J=1 TO 5
30     LET K=INT(6*RND+1)
40     PRINT K; " "
50   NEXT J
60   PRINT
70 PRINT
80 NEXT I
90 STOP
```

```
RUN
2 5 3 1 3
3 4 5 5 2
4 3 5 4 1
4 5 4 4 6
1 6 3 4 2
```

În linia 30 a programului am folosit instrucțiunea

```
30 LET K=INT (6*RND+1)
```

care are ca efect generarea numerelor întregi de la 1 la 6 inclusiv.

☐ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 220

Diferențele de scriere (v. specificațiile de programare și documentația de proiectare) apar în:

a) **Modul de introducere a matricei de date A:**

```
50 FOR Z=1 TO 6
51     PRINT C$ (Z); " : "
55     S=0
60     FOR I=1 TO 3
61       PRINT "R"; : PRINT USING "#"; I; : PRINT "  "; : INPUT A(Z,I)
65       S=S+A (Z, I)
70     NEXT I
75     B(Z)=S/3
80 NEXT Z
```

Remarcați că spre deosebire de programele precedente citirea datelor (v. formatul de editare "#")

```
61 PRINT "R"; : PRINT USING "#"; I; : PRINT "  "; : INPUT A (Z,I)
```

se realizează în aceeași buclă **FOR-NEXT** (v. liniile 60–70) pe care le și prelucrează.

b) **Modul de afișare a rezultatelor**

Sub capul de tabel tipărit cu instrucțiunea:

```
120 PRINT "ZIUA "; " R1", " R2", " R3", "MEDIA"
```

s-au afișat (v. instrucțiunile 140, 159, 160) livrările zilnice și media livrărilor calculată în variabila indexată B(Z).

```
140 FOR Z=1 TO 6
159    PRINT C$(Z), A(Z,1), A(Z,2), A(Z,3), B(Z)
160 NEXT Z
```

Pentru tipărirea mediei livrărilor pe rezervor, calculată în variabila indexată D(I) s-au scris instrucţiunile:

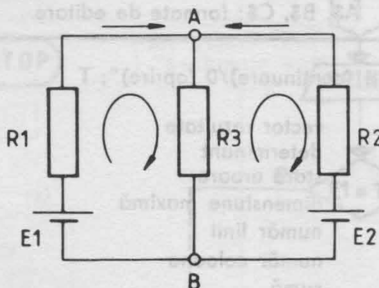
```
164 PRINT "MEDIA ↑ ";
165 FOR I=1 TO 3
167    PRINT D(I),
170 NEXT I
171 PRINT
```

c) **Modul de dimensionare a vectorului de date alfanumerice (nu este necesară declararea lungimii şirului de caractere).**

```
5 DIM C$(6), B(6), A(6,3), D(3)
```

Aplicaţii

1. Fie reţeaua electrică



în care se cunosc: E1=48 V; E2=8 V; R1=2Ω; R2=3Ω; R3=2Ω.

Să se determine intensitatea curentului prin fiecare ramură a reţelei.

Aplicînd legile lui Kirchoff, rezultă ecuaţiile:

$$I1+I2-I3=0; \quad (5)$$

$$I1R1+I3R3=E1; \quad (6)$$

$$I3R3+I2R2=E2 \quad (7)$$

Pentru rezolvarea sistemului de ecuaţii (5), (6), (7) aplicăm formulele lui Cramer.

$$I1=\frac{\begin{vmatrix} 0 & 1 & -1 \\ 48 & 0 & 2 \\ 8 & 3 & 2 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -1 \\ 2 & 0 & 2 \\ 0 & 3 & 2 \end{vmatrix}}=14 \text{ A}; \quad (8)$$

$$I2=\frac{\begin{vmatrix} 1 & 0 & -1 \\ 2 & 48 & 2 \\ 0 & 8 & 2 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -1 \\ 2 & 0 & 2 \\ 0 & 3 & 2 \end{vmatrix}}=-4 \text{ A}; \quad (9)$$

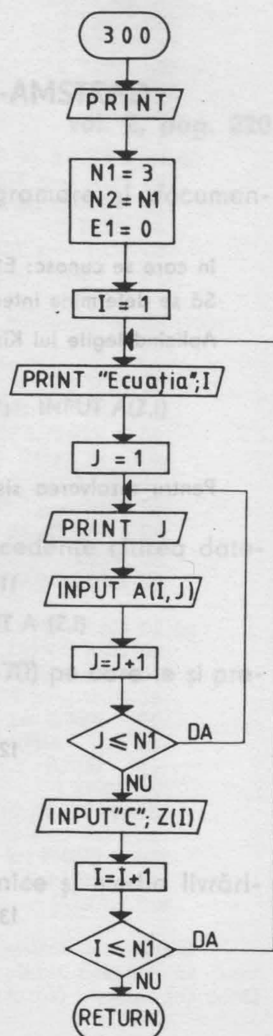
$$I3=\frac{\begin{vmatrix} 1 & 1 & 0 \\ 2 & 0 & 48 \\ 0 & 3 & 8 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -1 \\ 2 & 0 & 2 \\ 0 & 3 & 2 \end{vmatrix}}=10 \text{ A}. \quad (10)$$

În tabelul 6.4 şi figura 6.9 sînt prezentate tabela de variabile şi schema logică ale programului.

Tabelul 6.4

TABELA DE VARIABILE		
Variabile de intrare	Variabile de stare	Variabile de ieşire
A: matricea coeficienţilor Z: vector termeni liberi	M1: dimensiune maximă T: stare program (1/0) D3: determinant E1: stare eroare N1: număr linii N2: număr coloane S6: Sumă B: matrice de lucru A\$, B\$, C\$: formate de editare	C1: vector rezultate

```
5 INPUT "TASTAŢI 1 (continuare)/0 (oprire)"; T
10 WHILE T=1
15 REM    C1        vector rezultate
20 REM    D3        determinant
30 REM    E1        stare eroare
40 REM    M1        dimensiune maximă
50 REM    N1        număr linii
60 REM    N2        număr coloane
70 REM    S6        sumă
80 PRINT
90 A$="##.###^^^^"
95 B$="=##.###^^^^"
96 C$="##.##### "
97 M1=3
98 DIM Z(3), A(3,3), C1(3), B(3,3)
100 GOSUB 300
110 GOSUB 500
120 REM
130 PRINT "Matricea coeficienţilor"
140 FOR I=1 TO N1
150   FOR J=1 TO N2
160     PRINT USING A$; A (I, J);
170   NEXT J
180   PRINT USING B$; Z (I)
190 NEXT I
200 PRINT
210 IF (E1=1) THEN 280
220 PRINT "Soluţia sistemului"
230 PRINT
240 FOR I=1 TO N2
250   PRINT USING C$; C1(I);
260 NEXT I
270 PRINT
280 INPUT "TASTAŢI 1 (continuare)/0 (oprire)"; T
290 WEND
295 STOP
300 REM
```



```
310 PRINT
320 N1=3
330 N2=N1
340 E1=0
```

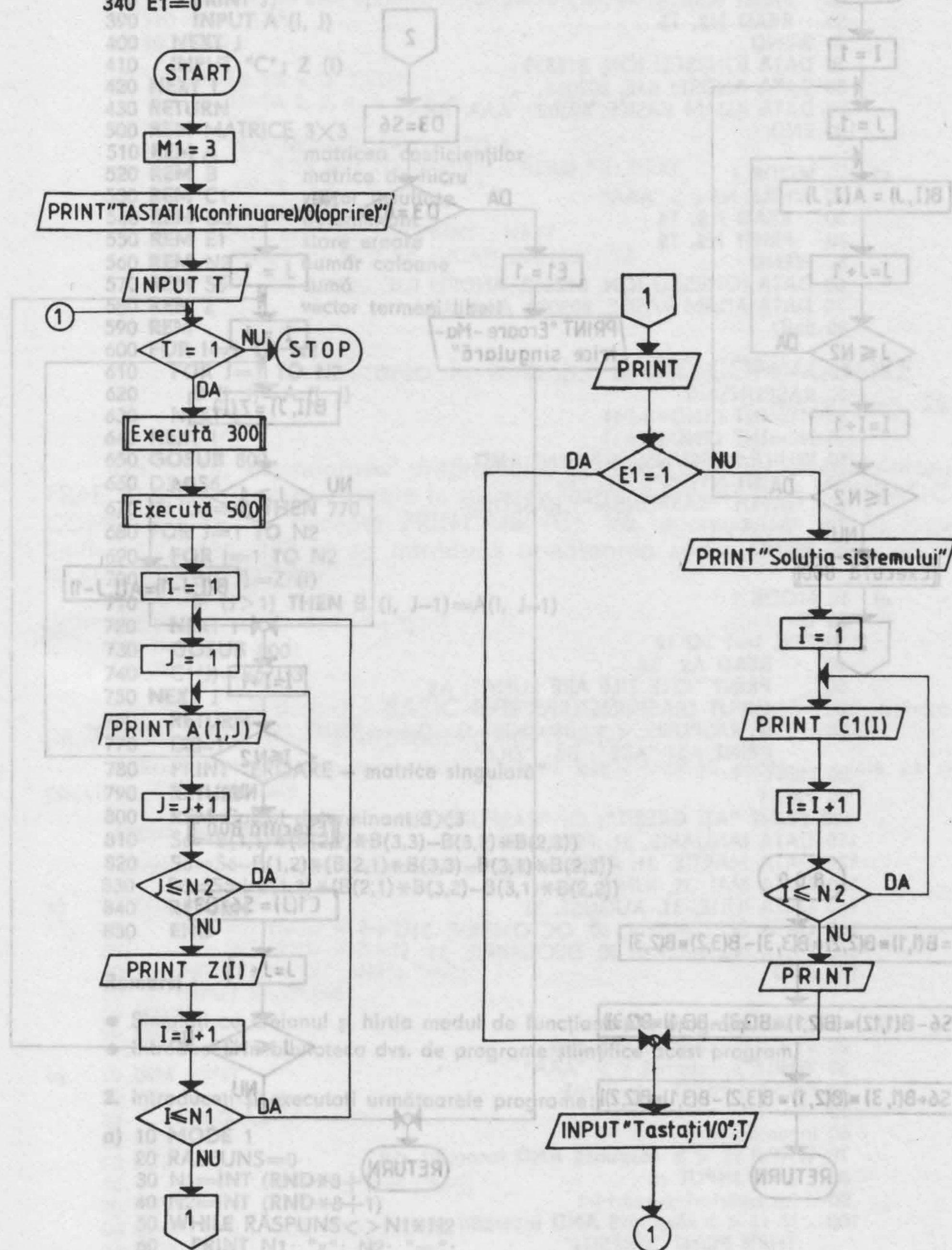


Fig. 6.9

500
I = 1
J = 1
B(I, J) = A(I, J)
J=J+1
J≤N2 DA
I=I+1
I≤N2 DA
NU
Execută 800
2

800
S6=B(1,1)*B(2,2)*B(3,3) – B(3,2)*B(2,3)
S6=S6 – B(1,12)*(B(2,1)*B(3,3) – B(3,1)*B(2,3))
S6=S6+B(1, 3)*(B(2,1)*B(3,2) – B(3,1)*B(2,2))
RETURN

2
D3=S6
DA D3 = 0 NU
E1 = 1
PRINT "Eroare - Matrice singulară"
J = 1
I = 1
B(I, J) = Z(I)
NU J > 1 DA
B(I, J–1)=A(I, J–1)
I=I+1
I≤N2 DA
NU
Execută 800
C1(J) = S6/D3
J=J+1
J ≤ N2 DA
NU
RETURN

Fig. 6.9

```
350 FOR I=1 TO N1
360   PRINT "Ecuaţia"; I
370   FOR J=1 TO N1
380     PRINT J; "   ";
390     INPUT A (I, J)
400   NEXT J
410   INPUT "C"; Z (I)
420 NEXT I
430 RETURN
500 REM MATRICE 3×3
510 REM A       matricea coeficienţilor
520 REM B       matrice de lucru
530 REM C1      vector rezultate
540 REM D3      determinant
550 REM E1      stare eroare
560 REM N2      număr coloane
570 REM S6      sumă
580 REM Z       vector termeni liberi
590 REM
600 FOR I=1 TO N2
610   FOR J=1 TO N2
620     B (I, J)=A (I, J)
630   NEXT J
640 NEXT I
650 GOSUB 800
660 D3=S6
670 IF (D3=0) THEN 770
680 FOR J=1 TO N2
690   FOR I=1 TO N2
700     B(I, J)=Z (I)
710     IF (J>1) THEN B (I, J–1)=A(I, J–1)
720   NEXT I
730   GOSUB 800
740   C1(J)=S6/D3
750 NEXT J
760 RETURN
770 E1=1
780 PRINT "EROARE – matrice singulară"
790 RETURN
800 REM Calcul determinant 3×3
810 S6=B(1,1)*(B(2,2)*B(3,3)–B(3,2)*B(2,3))
820 S6=S6–B(1,2)*(B(2,1)*B(3,3)–B(3,1)*B(2,3))
830 S6=S5+B(1,3)*(B(2,1)*B(3,2)–B(3,1)*B(2,2))
840 RETURN
850 END
```

Remarci

- Simulaţi cu creionul şi hîrtia modul de funcţionare a programului.
- Introduceţi în biblioteca dvs. de programe ştiinţifice acest program.

2. Introduceţi şi executaţi următoarele programe:

a)
```
10 MODE 1
20 RĂSPUNS=0
30 N1=INT (RND*8+1)
40 N2=INT (RND*8+1)
50 WHILE RĂSPUNS< >N1*N2
60   PRINT N1; "x"; N2; "=";
70   INPUT "RĂSPUNS="; RĂSPUNS
80 WEND
90 END
```

b)
```
10 MODE 1
20 READ N$, T$
30 WHILE N$<>"AAA"
40   PRINT N$, T$
50   READ N$, T$
60 WEND
70 DATA IONESCU ION, 812330
80 DATA ANDREI ILIE, 202015
90 DATA ADAM VASILE, 803020, AAA, AAA
100 END
```

c)
```
10 MODE 1
20 WHILE N$<>"AAA"
30   READ N$, T$
40   PRINT N$, T$
50 WEND
60 DATA IONESCU ION, 812330, ANDREI ILIE, 202015
70 DATA ADAM VASILE, 803020, AAA, AAA
80 END
```

d)
```
10 A=0
20 RĂSPUNS=0
30 N1=INT (RND*8+1)
40 N2=INT (RND*8+1)
50 WHILE RĂSPUNS<>N1*N2 AND A<4 '
60   PRINT N1; "x"; N2; "=";
70   INPUT "RĂSPUNS="; RĂSPUNS
75   A=A+1
80 WEND
```

e)
```
10 MODE 1
20 G=0
30 FOR I=1 TO 12
40      READ A$, B$
50      PRINT "CÎTE ZILE ARE LUNA"; A$
60      INPUT "RĂSPUNS" RĂSPUNS
70      IF RĂSPUNS <> B$ THEN G=G+1: PRINT "RĂSPUNS GREŞIT!" :
        PRINT A$; "ARE"; B$; "ZILE";
80 NEXT
90 PRINT
100 PRINT "AŢI GREŞIT"; G; "RĂSPUNSURI"
110 DATA IANUARIE, 31, FEBRUARIE, 28
120 DATA MARTIE, 31, APRILIE, 30
130 DATA MAI, 31, IUNIE, 30
140 DATA IULIE, 31, AUGUST, 31
150 DATA SEPTEMBRIE, 30, OCTOMBRIE, 31
160 DATA NOIEMBRIE, 30, DECEMBRIE, 31
170 END
```

f)
```
10 MODE 1
20 READ întrebare$, răspuns$
30 WHILE întrebare$ <> "AAA"
40 PRINT ; PRINT întrebare$
50 r$=""
60 încercări=1
70 WHILE r$ <> răspuns$ AND încercări <4
80   LINE INPUT r$ ,
90   încercări=încercări+1
100  IF r$ <> răspuns$ AND încercări <4
     THEN PRINT "GREŞIT!"
110 WEND
120 IF r$ <> răspuns$ THEN PRINT "Răspunsul este"; răspuns$
130 READ întrebare$, răspuns$
```

```
140 WEND
150 DATA Ce echipă românească a cîştigat CUPA CAMPIONILOR EUROPENI?,
    Steaua
160 DATA Care este antrenorul echipei RAPID, Schimbător
170 AAA, AAA
180 END
```

g)
```
 5 WHILE Y$ <> "NU"
10     DATA 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A
20     CLS
30     FOR NUMĂRCĂRŢI=1 TO 5
40         FOR I=1 TO RND(13) : READ P$: NEXT
50     PRINT TAB (15); P$; "  " ;
60     RESTORE : NEXT
70     FOR L=1 TO 15 : PRINT : NEXT
80     INPUT "DORIŢI SĂ JUCAŢI DA/NU"; Y$
90 WEND
```

☐ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 221

De notat, asemănarea programului cu cel executat pe calculatorul PRAE. Singura deosebire apare la scrierea instrucţiunii din linia 135 (BASIC-COMMODORE nu acceptă **PRINT USING**). Vă recomandăm ca în locul liniilor 130, 135, 138 să se introducă următoarea secvenţă de instrucţiuni BASIC-COMMODORE:

```
130 PRINT "A (" ; Z; " , " ; ")=" ;
140 INPUT A (Z, I)
142 PRINT
```

De notat că limbajul BASIC-COMMODORE nu acceptă nici instrucţiunile matriciale şi nici instrucţiunea **PRINT AT.**

Aplicaţie. Modificaţi programele precedente astfel încît să poată fi rulate pe un calculator COMMODORE.

Jocuri pe calculator

Introduceţi şi executaţi următoarele programe:

a)
```
10 FOR I=1 TO 10
20     N1=INT (RND * 8+1)
30     N2=INT (RND * 8+1)
40     PRINT N1 ; "X"; N2; "=";
50     INPUT RĂSPUNS
60 NEXT I
70 END
```

b)
```
10 DIM A$(4)
20 FOR I=1 TO 4
30     READ A$ (I)
40 NEXT I
50 FOR I=1 TO 4
60     PRINT "A$ (" ; I ; ")=" ; A$(I)
70 NEXT I
80 DATA UNU, DOI, TREI, PATRU
90 END
```

c)
```
10 INPUT "INTRODUCEŢI PRIMUL NUMĂR : " ; X
20 INPUT "INTRODUCEŢI AL DOILEA NUMĂR : " ; Y
```

```
30 PRINT "PRIMUL NUMĂR=" ; X
40 PRINT "AL DOILEA NUMĂR="; Y
50 PRINT "ALEGEŢI UNUL DIN OPERATORII: +, -, * /"
60 INPUT "OPŢIUNE: " ; R$
70 PRINT "OPŢIUNE: " ; R$
80 IF R$="+" THEN 140
90 IF R$="-" THEN 160
100 IF R$="*" THEN 180
110 IF R$=" / " THEN 200
120 PRINT "OPERATOR INCORECT"
130 GOTO 60
140 R1=X+Y
150 GO TO 250
160 R1=X-Y
170 GOTO 250
180 R1=A * B
190 GOTO 250
200 IF Y=0 THEN 230
210 R1=X/Y
220 GO TO 220
230 PRINT "LA ÎMPĂRŢIRE, Y TREBUIE SĂ FIE DIFERIT DE 0"
240 GO TO 10
250 INPUT "RĂSPUNSUL DVS= " ; R
260 PRINT "RĂSPUNSUL DVS= " ; R
270 IF R1<>R THEN 300
280 PRINT "FELICITĂRI!"
290 STOP
300 PRINT "RĂSPUNS GREŞIT"
310 GOTO 250
320 END
```

Observaţie. Programul vă ajută să vă dezvoltaţi calculul aritmetic (adunări, scăderi, înmulţiri, împărţiri) fără creion, gumă sau calculator.

☐ Particularităţi ale programării în limbajul BASIC-80

vol. 2, pag. 223

De notat (v. vol. 2, pag. 223) asemănarea programului cu cel executat pe calculatorul AMSTRAD. Remarcaţi şi în cadrul acestui program executat pe microcalculatoarele M 118, TPD, JUNIOR: a) modul de dimensionare a vectorului de date alfanumerice C$ (fără a se fi indicat lungimea elementelor sale); b) utilizarea instrucţiunii **PRINT USING** (v. linia 61).

Aplicaţie. Scrieţi un program BASIC-80 care:

a) înmulţeşte o matrice cu un scalar

```
100 FOR I=1 TO N
110     FOR J=1 TO M
120         Y (I, J)=X (I, J) * S
130     NEXT J
140 NEXT I
```

b) adună două matrice

```
100 FOR I=1 TO N
110     FOR J=1 TO M
120         Z(I,J)=X(I,J)+Y(I,J)
130     NEXT J
140 NEXT I
```

c) înmulţeşte două matrice:

$$X=\begin{bmatrix} x11 & x12 \\ x21 & x22 \\ x31 & x32 \end{bmatrix}$$

şi

$$Y=\begin{bmatrix} y11 & y12 & y13 \\ y21 & y22 & y23 \end{bmatrix}$$

$$Z=\begin{bmatrix} x11y11+x12y21 & x11y12+x12y22 & x11y13+x12y23 \\ x21y11+x22y21 & x21y12+x22y22 & x21y13+x22y23 \\ x31y11+x32y21 & x31y12+x32y22 & x31y13+x32y23 \end{bmatrix}$$

Fiecare element j_k al lui Z se formează din linia j a matricei X şi coloana *k* a matricei Y, conform relaţiei

$$Z_{jk}=X_{j1}\ Y_{1k}+X_{j2}\ Y_{2k}+\ldots+X_{jn}\ Y_{nk} \tag{11}$$

Pentru înmulţirea a două matrice vom avea nevoie să înmulţim transpusa unei matrici U cu matricea originală U.

De asemenea, va trebui să înmulţim vectorul Y cu matricea U.

Programul care urmează efectuează ambele operaţii. Secvenţa de instrucţiuni 70÷150 generează matricea U şi vectorul Z. Secvenţa de instrucţiuni 160–290 calculează matricea A şi vectorul Z de care avem nevoie, conform ecuaţiei

$$U^TU=A \tag{12}$$

şi

$$YU=Z \tag{13}$$

În acest caz, matricea U conţine 5 linii şi 3 coloane. Vectorul Y conţine cinci elemente.

Programul atribuie lui N1 şi N2 valoarea 5 respectiv 3, dimensionează pe Z, A, Y, U (vezi linia 50) şi defineşte şabloanele de editare A$ şi B$ din liniile 10 şi 20.

Secvenţa de instrucţiuni 160÷240 merită să fie studiată cu atenţie. Ea conţine trei bucle nested. Bucla din centru (190÷220) realizează înmulţirea propriu-zisă a matricelor, sau mai exact înmulţirea unei matrice cu transpusa ei.

Buclele 170÷230 şi 250÷270 sînt de nivelul al doilea. Linia 170 ţine cont că matricea A va fi simetrică. În acest caz trebuie calculate numai elementele diagonalei de deasupra. Elementele de sub diagonală sînt atribuite de către linia 210. Vor fi termeni ca:

$$A(1, 3)=A(3, 1)$$

În sfîrşit, bucla din liniile 250÷270 realizează înmulţirea Y U.

Remarcaţi linia 180:

180 A(K, L)=0

care iniţializează cu zero fiecare element al lui A înainte de a începe operaţia de adunare din linia 200. Matricea A va avea trei linii (la fel ca transpusa matricei U) şi trei coloane (la fel ca matricea U). Vectorul Z va avea lungimea 3. De fapt, atît Y cît şi Z trebuie consideraţi ca vectori linie. În mod alternativ, dacă vrem să considerăm vectorii Y şi Z ca vectori coloană, atunci va trebui să scriem ecuaţia de înmulţire ca:

$$Y^TU=Z^T$$

unde vectorii-coloană transpuşi devin vectori linie.

```
10 A$=" ####.# "
20 B$="=####.# "
30 N1=5
40 N2=3
50 DIM Z(3), A(3, 3), Y(5), U(5, 3)
60 REM Citire date
70 PRINT
80 FOR I=1 TO N1
90     U(I, 1)=1
```

```
100   FOR J=2 TO N2
110        U(I,J)=I * U(I,J-1)
120   NEXT J
130   Y(I)=2 * I
140 NEXT I
150 REM
160 FOR K=1 TO N2
170   FOR L=1 TO K
180      A(K, L)=0
190      FOR I=1 TO N1
200         A(K, L)=A(K, L)+U(I, L) * U(I, K)
210         IF (K< >L) THEN A(L, K)=A(K, L)
220      NEXT I
230   NEXT L
240   Z(K)=0
250   FOR I=1 TO N1
260      Z(K)=Z(K)+Y(I) * U (I, K)
270   NEXT I
280 NEXT K
290 PRINT
300 PRINT "   U    Y"
310 FOR I=1 TO N1
320 FOR J=1 TO N2
330      PRINT USING A$; U(I%, J%);
340   NEXT J
350   PRINT USING B$; Y(I)
360 NEXT I
370 PRINT
380 PRINT "   A    Z"
390 FOR I=1 TO N2
400   FOR J=1 TO N2
410      PRINT USING A$; A(I,J)
420   NEXT J
430   PRINT USING B$; Z(I)
440 NEXT I
450 PRINT
460 END
```

d) calculează determinantul unei matrice 3 pe 3.

```
10 REM DETERMINANTUL UNEI MATRICE 3×3
20 REM  N1  număr linii
30 REM  N2  număr coloane
40 REM
50 A$="###·###"
60 N1=3
70 N2=3
80 DIM B(3, 3)
90 REM Introducere date
100 PRINT
110 FOR I=1 TO N1
120    PRINT "LINIA" ; I
130    FOR J=1 TO N2
140       PRINT J ; " " ;
150       INPUT B (I, J)
160    NEXT J
170 NEXT I
180 REM CALCUL DETERMINANT – REGULA LUI CRAMER
190 S=B(1,1) * (B(2,2) * B(3,3)-B(3,2) * B(2,3))
200 S=S-B(1,2) * (B(2,1) * B(3,3)-B(3,1) * B(2,3))
210 S=S+B(1,3) * (B(2,1) * B(3,2)-B(3,1) * B(2,2))
220 REM
230 PRINT
240 FOR I=1 TO N1
```

```
250     FOR J=1 TO N2
260          PRINT USING A$ ; B(I,J) ;
270     NEXT J
275 PRINT
280 NEXT I
290 PRINT
300 PRINT "DETERMINANTUL = "; S
310 PRINT
320 END
```

Definirea unei funcţii scrise de utilizator

În foarte multe aplicaţii intervin funcţii cu unul sau mai mulţi parametri. Limbajul BASIC-80 oferă utilizatorului facilitatea definirii acestor funcţii în cadrul programului cu instrucţiunea **DEF FN**. Formatul general al instrucţiunii este:

Format general
DEF FN ⟨nume⟩ [⟨listă-parametri⟩] = ⟨def-funcţie⟩

unde:

⟨nume⟩ este numele unei variabile BASIC-80;
⟨listă-parametri⟩ conţine variabilele utilizate la definirea funcţiei (parametrii trebuie separaţi prin virgule);
⟨def-funcţie⟩ este o expresie.

Aplicaţie. Să se calculeze

$$\int_1^8 \frac{dx}{x}$$

prin metoda trapezelor. Se va utiliza subrutina TRAPEZ de integrare prin metoda trapezelor.
La proiectarea şi realizarea programului se vor avea în vedere următoarele funcţiuni:

1. Citirea/afişarea datelor de intrare
2. Apelarea subrutinei TRAPEZ (500)
3. Calculul integralei
4. Imprimarea rezultatului

În tabelul 6.5 şi tabelul 6.6 sînt ilustrate tabela de variabile şi programul sursă al subrutinei.

Tabelul 6.5

TABELA DE VARIABILE		
Variabile de intrare	Variabile de stare	Variabile de ieşire
A: limita inferioară	H: pasul de integrare	TI: valoarea integralei
B: limita superioară		
N: numărul de intervale		

Tabelul 6.6

PROGRAM SURSĂ
500 H=(B–A)/N 510 TI=(F(A)+F(B))/2 520 **FOR** K=1 **TO** N–1 530 TI=TI+F(A+K * H) 540 **NEXT** K 550 TI=TI * H 560 **RETURN**

Remarcă. Introduceți în biblioteca dvs. de programe științifice subrutina TRAPEZ.

```
  5 DEF FN F (X)=1/X
 10 INPUT "Număr intervale" ; N
 20 INPUT "Limite integrare" ; A, B
 30 GOSUB 500
 40 PRINT "Aria="; TI
 50 STOP
500 (v. subrutina TRAPEZ)
 .
 .
 .
560
570 END
```

Aplicații

1. Introduceți și executați următorul program:

```
10  RANDOMIZE                RUN
20  FOR I=1 TO 4             0. 5199228
30    PRINT RND              0. 3434087
40  NEXT I                   0. 2987809
                             0. 7020067
```

Observație. Funcția **RANDOMIZE** inițializează generatorul de numere aleatoare. Ea are formatul:

Format general
RANDOMIZE ⟨expresie⟩

Dacă **RANDOMIZE** lipsește, funcția **RND** va da aceeași secvență de numere aleatoare la fiecare execuție a programului.

2. Scrieți un program care să simuleze aruncarea unei monezi de 30 de ori. Programul va contoriza și afișa de cîte ori "cade" leul sau stema.

```
 5 REM L : LEUL ; S : STEMA
10 L=0
20 S=0
30 FOR I=1 TO 30
40   A=INT (2 * RND+1)
50   IF A=2 THEN L=L+1 ELSE S=S+1
60 NEXT I
70 PRINT TAB 8 ; " L=" ; L; TAB 20 ; "S="; S
80 END
```

3. Scrieți un program care să simuleze aruncarea a două zaruri de 80 de ori. Programul va afișa numărul și frecvența de apariție a unei fețe a zarului sub forma unei diagrame.

Exemplu:

NUMĂR	FRECVENȚA	DIAGRAMA
2	2	**
12	4	****
.	.	.
.	.	.
.	.	.

```
10 DIM S(15)
20 FOR I=2 TO 12
30    S(I)=0
40 NEXT I
50 FOR I=1 TO 80
60    S1=INT (6 * RND+1)
70    S2=INT (6 * RND+1)
80    S=S1+S2
90    S(S)=S(S)+1
100 NEXT I
110   PRINT
120 FOR I=2 TO 12
130   PRINT I; TAB 5; S(I); TAB 10;
140   IF S(I)=0 THEN 190
150   FOR J=1 TO S(J)
160     PRINT " * " ;
170   NEXT J
180   PRINT
190 NEXT I
200 STOP
```

4. Să se rezolve sistemul de ecuații (v. aplicație, pag. 319) prin metoda eliminării a lui Gauss.

Tabela de variabile și schema logică ale programului (subrutinele 500; 5000 sînt prezentate în tabelul 6.7 și figura 6.10).

Tabelul 6.7

TABELA DE VARIABILE				
Variabile de intrare	Variabile de stare		Variabile de ieșire	
N1: număr linii (ecuații)	E1:	stare	C1:	vectorul rezultatelor
	M1:	lungimea maximă		
N2: număr coloane	B:	matrice de lucru		
	B1:	matrice de lucru		
A: matricea coeficienților	H1:	variabila de lucru		
Z: vector termeni liberi	W1:	vector de lucru		
	S6:	suma		

Programul debutează prin a vă întreba dacă doriți sau nu continuarea. Răspundeți la această întrebare prin a tasta 1 pentru continuare și 0 pentru oprire. Următoarea întrebare se referă la numărul de ecuații ale sistemului. Răspundeți cu un număr între 2 și 8. Coeficienții necunoscutelor și termenii liberi sînt introduși pe rînd pentru fiecare ecuație în parte. Introduceți valorile pentru sistemul de ecuații studiat (v. aplicația de la pag. 319) și verificați dacă vectorul soluție este 14, –4, 10.

Programul imprimă un mesaj de eroare atunci cînd matricea coeficienților este singulară:

EROARE – matrice singulară!

Programul poate fi întrerupt în două rînduri: a) tastînd 0 pentru variabila T; b) introducînd o valoare sub 2 pentru numărul de ecuații (N1).

În sfîrșit, programul apelează două subrutine: 500 (introducere date) și 5000 (rezolvarea propriu-zisă a sistemului).

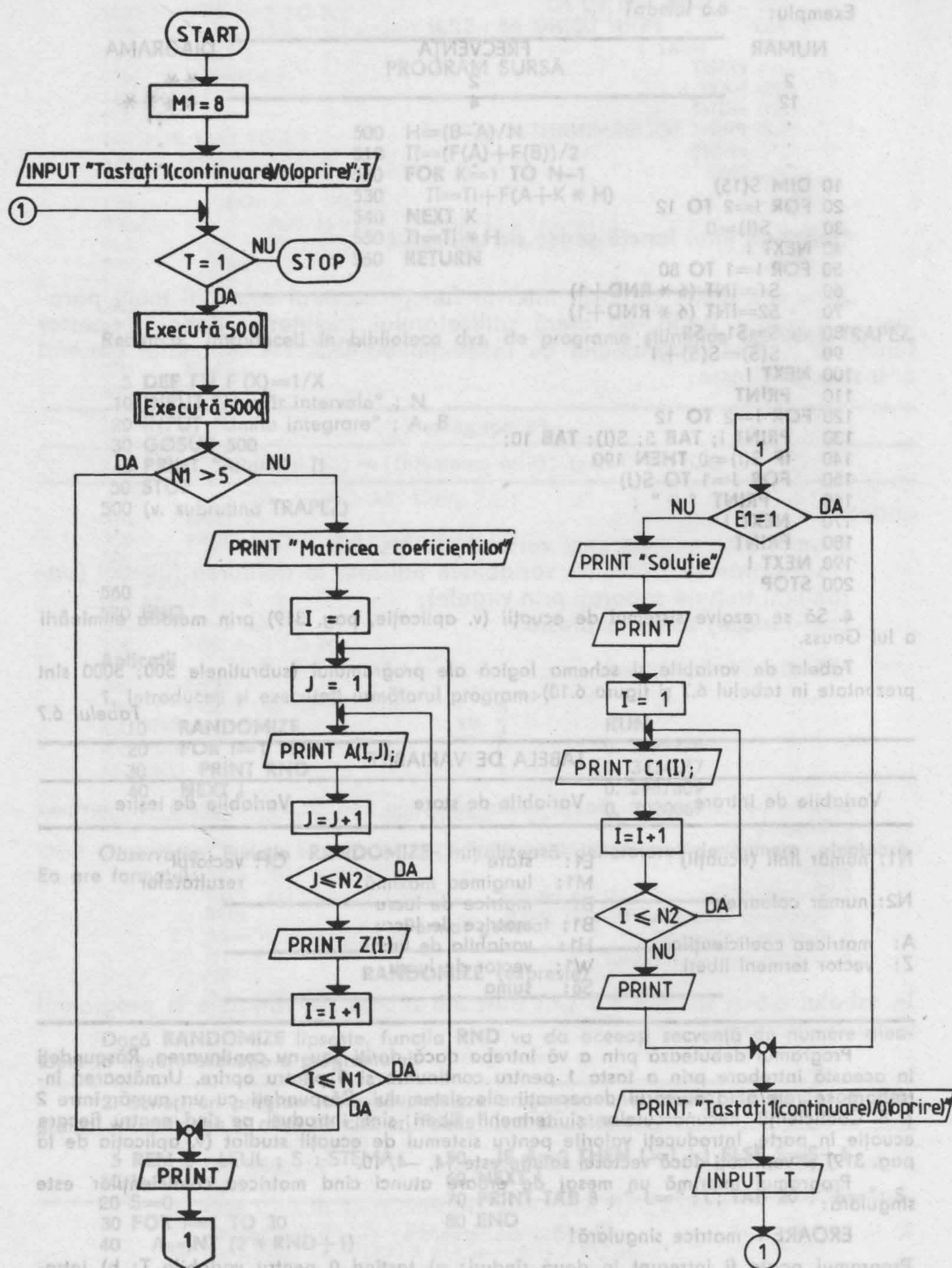
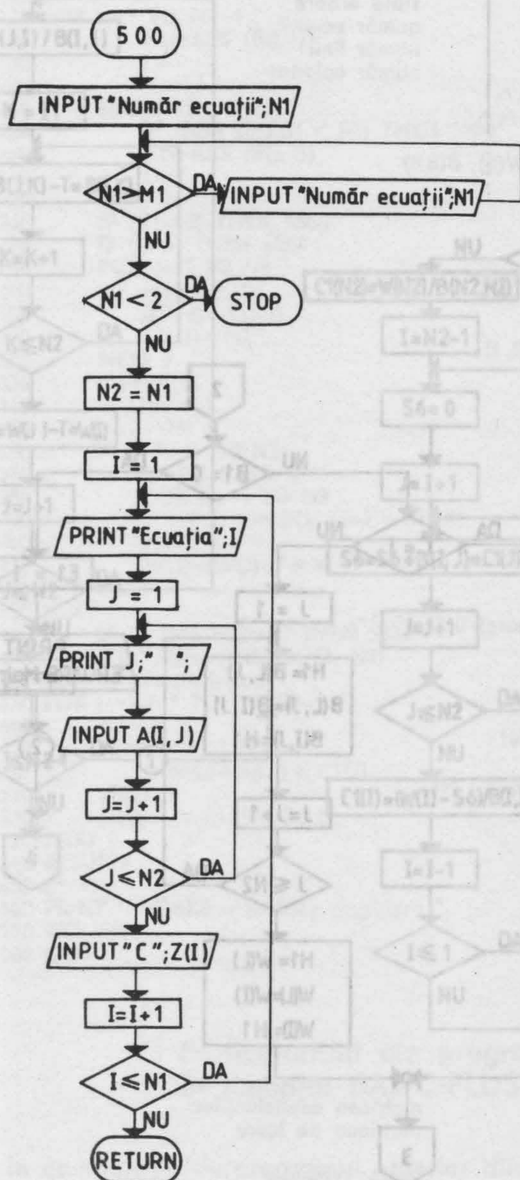

START
M1=8
INPUT "Tastaţi 1(continuare)/0(oprire)";T
1
T = 1
NU
STOP
DA
Execută 500
Execută 5000
DA
N1 >5
NU
PRINT "Matricea coeficienţilor"
I = 1
J = 1
PRINT A(I,J);
J=J+1
J≤N2
DA
PRINT Z(I)
I=I+1
I≤N1
DA
DA
PRINT
1
1
NU
E1=1
DA
PRINT "Soluţie"
PRINT
I = 1
PRINT C1(I);
I=I+1
I ≤ N2
DA
NU
PRINT
PRINT "Tastaţi 1(continuare)/0(oprire)"
INPUT T
1

Fig. 6.10.


500
INPUT "Număr ecuaţii";N1
N1>M1
DA
INPUT "Număr ecuaţii";N1
NU
N1 < 2
DA
STOP
NU
N2 = N1
I = 1
PRINT "Ecuaţia";I
J = 1
PRINT J;" ";
INPUT A(I, J)
J=J+1
J≤N2
DA
NU
INPUT " C ";Z(I)
I=I+1
I≤N1
DA
NU
RETURN

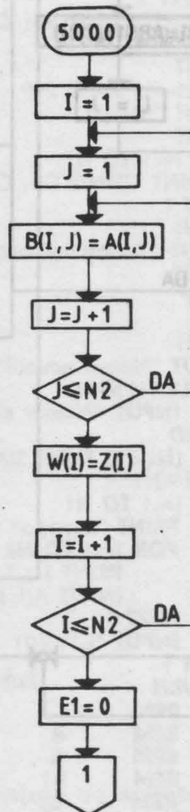

5000
I = 1
J = 1
B(I, J) = A(I, J)
J=J+1
J≤N2
DA
W(I)=Z(I)
I=I+1
I≤N2
DA
E1=0
1

Fig. 6.10

1

I = 1

①

B1 = ABS(B(I,I))

L = I

I1 = I + 1

J = I1

DA ABS(B(J,I)) < B1 NU

B1 = ABS(B(J,I))

L = J

J = J+1

J ≤ N2 DA

NU

2

2

NU B1 = 0 DA

DA L = I NU

J = 1

H1 = B(L,J)
B(L,J) = B(I,J)
B(I,J) = H1

J = J + 1

J ≤ N2 DA

H1 = W(L)
W(L) = W(I)
W(I) = H1

3

E1 = 1

PRINT
"EROARE-Matrice singulară"

②

Fig. 6.10

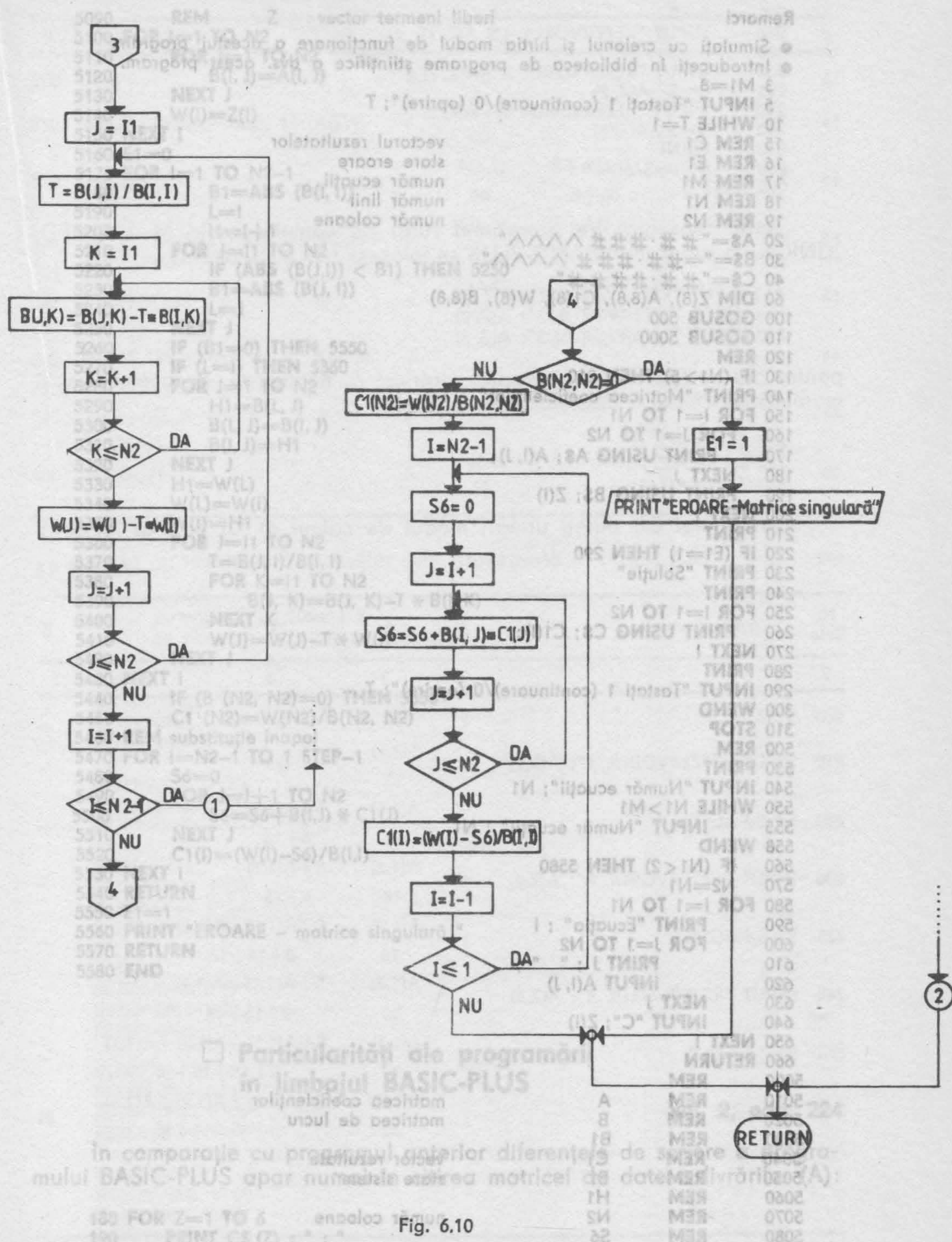

3
J = I1
T = B(J,I) / B(I,I)
K = I1
B(J,K) = B(J,K) - T * B(I,K)
K=K+1
K≤N2
DA
W(J) = W(J) - T * W(I)
J=J+1
J≤N2
DA
NU
I=I+1
I≤N2-1
DA
NU
1
4
4
B(N2,N2)=0
NU
DA
C1(N2)=W(N2)/B(N2,N2)
I=N2-1
S6= 0
J=I+1
S6=S6+B(I,J)*C1(J)
J=J+1
J≤N2
DA
NU
C1(I)=(W(I)-S6)/B(I,I)
I=I-1
I≤1
DA
NU
E1 = 1
PRINT "EROARE-Matrice singulară"
2
RETURN

Fig. 6.10

Remarci

- Simulaţi cu creionul şi hirtia modul de funcţionare a acestui program.
- Introduceţi în biblioteca de programe ştiinţifice a dvs. acest program.

```
   3 M1=8
   5 INPUT "Tastaţi 1 (continuare)/0 (oprire)"; T
  10 WHILE T=1
  15 REM C1                              vectorul rezultatelor
  16 REM E1                              stare eroare
  17 REM M1                              număr ecuaţii
  18 REM N1                              număr linii
  19 REM N2                              număr coloane
  20 A$="##.### ^^^^"
  30 B$="=##.### ^^^^"
  40 C$="##.####"
  60 DIM Z(8), A(8,8), C1(8), W(8), B(8,8)
 100 GOSUB 500
 110 GOSUB 5000
 120 REM
 130 IF (N1>5) THEN 210
 140 PRINT "Matricea coeficienţilor"
 150 FOR I=1 TO N1
 160   FOR J=1 TO N2
 170     PRINT USING A$; A(I, J);
 180   NEXT J
 190   PRINT USING B$; Z(I)
 200 NEXT I
 210 PRINT
 220 IF (E1=1) THEN 290
 230 PRINT "Soluţie"
 240 PRINT
 250 FOR I=1 TO N2
 260   PRINT USING C$; C1(I);
 270 NEXT I
 280 PRINT
 290 INPUT "Tastaţi 1 (continuare)/0 (oprire)"; T
 300 WEND
 310 STOP
 500 REM
 530 PRINT
 540 INPUT "Număr ecuaţii"; N1
 550 WHILE N1>M1
 555       INPUT "Număr ecuaţii" ; N1
 558 WEND
 560   IF (N1<2) THEN 5580
 570   N2=N1
 580 FOR I=1 TO N1
 590       PRINT "Ecuaţia" ; I
 600       FOR J=1 TO N2
 610             PRINT J ; "   ";
 620             INPUT A(I, J)
 630       NEXT J
 640       INPUT "C"; Z(I)
 650 NEXT I
 660 RETURN
5000       REM
5010       REM      A               matricea coeficienţilor
5020       REM      B               matricea de lucru
5030       REM      B1
5040       REM      C1              vector rezultate
5050       REM      E1              stare sistem
5060       REM      H1
5070       REM      N2              număr coloane
5080       REM      S6
```

```
5090       REM       Z       vector termeni liberi
5100 FOR I=1 TO N2
5110       FOR J=1 TO N2
5120              B(I, J)=A(I, J)
5130       NEXT J
5140       W(I)=Z(I)
5150 NEXT I
5160 E1=0
5170 FOR I=1 TO N2-1
5180              B1=ABS (B(I, I))
5190              L=I
5200              I1=I+1
5210       FOR J=I1 TO N2
5220              IF (ABS (B(J,I)) < B1) THEN 5250
5230              B1=ABS (B(J, I))
5240              L=J
5250       NEXT J
5260       IF (B1=0) THEN 5550
5270       IF (L=I) THEN 5360
5280       FOR J=1 TO N2
5290              H1=B(L, J)
5300              B(L, J)=B(I, J)
5310              B(I, J)=H1
5320       NEXT J
5330       H1=W(L)
5340       W(L)=W(I)
5350       W(I)=H1
5360       FOR J=I1 TO N2
5370              T=B(J, I)/B(I, I)
5380              FOR K=I1 TO N2
5390                     B(J, K)=B(J, K)-T * B(I, K)
5400              NEXT K
5410              W(J)=W(J)-T* W(I)
5420       NEXT J
5430 NEXT I
5440       IF (B (N2, N2)=0) THEN 5550
5450       C1 (N2)=W(N2)/B(N2, N2)
5460 REM substituţie înapoi
5470 FOR I=N2-1 TO 1 STEP-1
5480       S6=0
5490       FOR J=I+1 TO N2
5500              S6=S6+B(I,J) * C1(J)
5510       NEXT J
5520       C1(I)=(W(I)-S6)/B(I,I)
5530 NEXT I
5540 RETURN
5550 E1=1
5560 PRINT "EROARE - matrice singulară!"
5570 RETURN
5580 END
```

☐ Particularităţi ale programării în limbajul BASIC-PLUS

vol. 2, pag. 224

În comparaţie cu programul anterior diferenţele de scriere a programului BASIC-PLUS apar numai în citirea matricei de date a livrărilor (A):

```
180 FOR Z=1 TO 6
190       PRINT C$ (Z) ; " : "
```

```
200 S=0
210 INPUT "REZERVORUL 1"; A(Z, 1)
220 S=S+A(Z,1)
230 INPUT "REZERVORUL 2"; A(Z, 2)
235 S=S+A(Z, 2)
240 INPUT "REZERVORUL 3"; A(Z, 3)
260 S=S+A(Z, 3)
270 PRINT
280 B(Z)=S/3
290 NEXT Z
```

Spre deosebire de programul rulat pe calculatoarele M 118, TPD, JUNIOR, în programul BASIC-PLUS se folosesc trei instrucțiuni:

```
210 INPUT "REZERVORUL 1"; A(Z, 1)
230 INPUT "REZERVORUL 2"; A(Z, 2)
240 INPUT "REZERVORUL 3"; A(Z, 3)
```

pentru citirea (pe linie) a valorilor matricei A. Remarcați, de asemenea, și maniera de calcul a totalului general livrări:

```
220 S=S+A(Z, 1)
235 S=S+A(Z, 2)
260 S=S+A(Z, 3)
```

În tabelul 6.8 puteți urmări modul de calcul al lui S și al lui B(Z).

Tabelul 6.8

Nr. linie	Instrucțiune	Z	A	S	B
0	1	2	3	4	5
200	S=0	1	–	0	–
210	**INPUT** "REZERVORUL 1"; A(Z,1)	1	A(1,1) 15	0	–
220	S=S+A(Z,1)	1	A(1,1) 15	S+A(1,1) 0+15	–
230	**INPUT** "REZERVORUL 2"; A(Z,2)	1	A(1,2) 18	S+A(1,1) 0+15	–
235	S=S+A(Z,2)	1	A(1,2) 18	S+A(1,1)+A(1,2) 0+15+18	–
240	**INPUT** "REZERVORUL 3"; A(Z,3)	1	A(1,3) 30	S+A(1,1)+A(1,2) 0+15+18	–
260	S=S+A(Z,3)	1	A(1,3) 30	S+A(1,1)+A(1,2)+A(1,3) 0+15+18+30	–
280	B(Z)=S/3	1	A(1,3) 30	S+A(1,1)+A(1,2)+A(1,3) 0+15+18+30	21
⋮	⋮	⋮	⋮	⋮	⋮

0	1	2	3	4	5
200	S=0	6	0	0	44
210	**INPUT** "REZERVORUL 1"; A(Z,1)	6	A(6,1) 60	0	44
220	S=S+A(Z,1)	6	A(6,1) 60	S+A(6,1) 0+60	44
230	**INPUT** "REZERVORUL 2"; A(Z,2)	6	A(6,2) 60	S+A(6,1) 60	44
235	S=S+A(Z,2)	6	A(6,2) 60	S+A(6,1)+A(6,2) 0+60+60	44
240	**INPUT** "REZERVORUL 3"; A(Z, 3)	6	A(6,3) 60	S+A(6,1)+A(6,2) 0+60+60	44
260	S=S+A(Z,3)	6	A(6,3) 60	S+A(6,1)+A(6,2)+A(6,3) 0+60+60	44
280	B(Z)=S/3	6	A(6,3) 60	S+A(6,1)+A(6,2)+A(6,3) 0+60+60+60	60

Lista cu rezultatele execuţiei este prezentată în figura 6.11.

```
:RUN
          INTRODUCETI LIVRARILE
          DE BENZINA PE ZILE IN ORDINEA
             R1       R2       R3
LUNI :
REZERVORUL 1?15
REZERVORUL 2?18
REZERVORUL 3?30

REZERVORUL 1?23
REZERVORUL 2?40
REZERVORUL 3?0

MIERCURI :
REZERVORUL 1?55
REZERVORUL 2?26
REZERVORUL 3?1

JOI :
REZERVORUL 1?0
REZERVORUL 2?0
REZERVORUL 3?0

VINERI :
REZERVORUL 1?12
REZERVORUL 2?60
REZERVORUL 3?60

SIMBATA :
REZERVORUL 1?60
REZERVORUL 2?60
REZERVORUL 3?60
```

Fig. 6.11.

```
ZIUA        R1       R2        R3        MEDIA
--------------------------------------------------------
LUNI        15       18        30        21
MARTI       23       40        0         21
MIERCURI    55       26        1         27.3333
JOI         0        0         25        8.33333
VINERI      12       60        60        44
SIMBATA     60       60        60        60
--------------------------------------------------------
MEDIA       27.5     34        29.3333

MAXIMUL LIVRARILOR  60  TONE
LA REZERVORUL     = 2
IN ZIUA DE        = VINERI

STOP AT LINE 710
```

Fig. 6.11. continuare.

Observație. Pentru citirea dinamică a matricei A se poate utiliza și instrucțiunea **MAT INPUT.**

Aplicație. Să se determine prin metoda Newton–Raphson soluția ecuației:

$$x^2-3=0$$

cunoscînd: soluția inițială x=3, precizia=0,000001.

Programul va apela subrutina NERA (5000) a cărei tabelă de variabile este ilustrată în tabelul 6.9. Programul sursă al subrutinei este prezentat în tabelul 6.10.

Tabelul 6.9

TABELA DE VARIABILE

Variabile de intrare		Variabile de stare		Variabile de ieșire	
T1:	precizia	D6:	delta X	X1:	soluția ecuației
X:	soluția inițială	F:	funcția		
		F1:	derivata funcției		

Tabelul 6.10

PROGRAM SURSĂ

```
5000 REM                    5060 D6=F/F1
5010 X1=X                   5070 X=X1-D6
5020 GOSUB 5200             5080 PRINT "X="; X1; ", fX="; F; ", dfx="; F1
5030 REM                    5090 IF (ABS(D6)>=ABS(T1 * X)) THEN 5010
5040 REM                    5100 RETURN
5050 REM
```

Notă. Subrutina 5200 conține funcția (F) și derivata funcției. De exemplu, pentru ecuația $x^2-3=0$, corespund instrucțiunile:

```
5200 F=X * X-3
5210 F1=2 * X
5220 RETURN
```

```
10 REM Metoda Newton-Raphson
15 REM X1                        Soluţia iniţială
20 REM D6                        Delta x
30 REM F                         Funcţia
40 REM F1                        Derivata funcţiei
50 REM T1                        Precizia
55 REM X                         Soluţia ecuaţiei
60 REM
70 T1=.000001
80 X=3
90 GOSUB 5000
100 PRINT
110 PRINT "Soluţia este"; X
120 STOP
5000 REM
5010 X1=X
5020 GOSUB 5200
5030 REM
5040 REM
5050 REM
5060 D6=F/F1
5070 X=X1-D6
5080 PRINT "X="; X1; ", f x="; F; ", d fx="; F1
5090 IF (ABS(D6)>=ABS (T1 * X)) THEN 5010
5100 RETURN
5200 F=X * X-3
5210 F1=2 * X
5220 RETURN
5230 END
```

TEST

Scrieţi un program BASIC-PLUS care să editeze matricea:

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

R.

```
10 DIM X(5, 3)
20 MAT READ X
30 MAT PRINT X ;
40 DATA 1, 2, 3, 4, 5, 6, 7
50 DATA 8, 9, 10, 11, 12, 13, 14, 15
60 END
```

Precizaţi care va fi rezultatul execuţiei programului, dacă linia 10 se modifică în:
10 DIM X(3, 5)

Jocuri pe calculator

1. Scrieţi un program care să simuleze aruncarea a două zaruri de 50 de ori. Programul va afişa numărul şi frecvenţa de apariţie a unei feţe a celor două zaruri.

```
10 DIM S(15)
20 FOR I=2 TO 12
30   LET S(I)=0
40 NEXT I
50 FOR I=1 TO 50
60   LET S1=INT (6 * RND+1)
70   LET S2=INT (6 * RND+1)
80   LET S=S1+S2
90   LET S(S)=S(S)+1
100 NEXT I
110 PRINT
120 PRINT "NUMĂR"; "FRECVENŢA"
```

```
130 FOR I=2 TO 12
140   PRINT I, S(I)
150 NEXT I
160 STOP
```

2. Scrieți un program care să simuleze aruncarea unui zar de 50 de ori. Programul va afișa numărul și frecvența de apariție a unei fețe a zarului.

```
10 DIM S(6)
20 FOR I=1 TO 6
30   S(I)=0
40 NEXT I
50 FOR I=1 TO 50
60   S=INT (6 * RND+1)
70   S(S)=S(S)+1
80 NEXT I
90 PRINT
100 PRINT "NUMĂR"; "FRECVENȚA"
110 FOR I=1 TO 6
120   PRINT I, S(I)
130 NEXT I
140 STOP
```

☐ Particularități ale programării în limbajul ABASIC

vol. 2, pag. 226

De notat (v. vol. 2, pag. 226) asemănarea programului cu cele executate pe calculatoarele AMSTRAD, M118, TPD și JUNIOR.

La o analiză mai puțin atentă a programului grăbindu-vă poate să treceți mai repede la tema acestei conversații, s-ar putea să rămînă neobservat un amănunt important legat de instrucțiunea **DATA.**

Comparînd-o cu instrucțiunea omoloagă se constată că în ABASIC constantele alfanumerice (șir) trebuie să fie cuprinse între ghilimele.

TEST

Să se determine prin metoda Newton–Raphson soluția ecuației

$$x^2-3=0$$

cunoscînd: soluția inițială x=3, precizia=0,000001 și numărul de iterații 20. Programul va apela subrutina NERA1. Programul sursă al subrutinei este prezentat în tabelul 6.11 iar tabela de variabile a aceleași subrutine este ilustrată în tabelul 6.12.

Tabelul 6.11

PROGRAM SURSA

```
5000 REM
5010 E1%=FO%
5020 FOR I%=1 TO M5%
5030   X1=X
5040   GOSUB 5200
5050   IF (ABS (FA)>H2) THEN 5090
5060   PRINT "EROARE"
5070   E1%=T0%
5080   GOTO 5160
5090   D6=F/F1
5100   X=X1-D6
5110   PRINT "X="; X1; ", fx="; ", dfx="; F1
5120   IF (ABS (D6)<=ABS (T1 * X)) THEN 5160
5130 NEXT I%
5140 PRINT "LIPSA CONVERGENTĂ IN"; M5%; "ITERATII"
5150 E1%=T0%
5160 RETURN

5200 F=X * X-3
5210 F1=2 * X
5220 RETURN
```

Tabelul 6.12

TABELA DE VARIABILE

Variabile de intrare		Variabile de stare		Variabile de ieşire	
X:	soluţie iniţială	H2:	număr foarte mic (1E–15)	X1:	soluţia ecuaţiei
T1:	precizia	F:	funcţia		
M5%:	număr de iteraţii	F1:	derivata funcţiei		
		D6:	delta X		
		FO%:	FALSE		
		T0%:	TRUE		
		I:	variabila de control a buclei		
		E1%:	stare eroare		

Remarcă. Introduceţi în biblioteca dvs. de programe ştiinţifice subrutina NERA1.

```
10 REM D6        Delta X
15 REM E1        stare eroare
16 REM F         funcţia
17 REM F1        derivaţia funcţiei
18 REM FO        FALS (zero)
19 REM H2        număr mic
20 REM M5        număr iteraţii
21 REM T0        ADEVĂRAT (unu)
22 REM T1        precizia
23 T1=0.000001
30 H2=1E-15
40 M5%=20
50 FO%=0
60 T0%=NOT FO%
70 INPUT "Soluţia iniţială"; X
80 IF (X<-19) THEN 9999
90 GOSUB 5000
100 PRINT
110 IF (E1%=FO%) THEN PRINT "Soluţia este"; X
120 GO TO 70
5000
   (v. subrutina NERA1)
5200 F=X * X
5210 F1=2 * X
5220 RETURN
9999 END
```

Aplicaţii

1. Scrieţi un program care să furnizeze, la cerere, capitala unei ţări:

```
10 INPUT "ŢARA"; P$
30 WHILE P$ <> A$ AND P$ <> "XXX"
40    READ A$, B$
50    IF P$=A$ THEN PRINT P$, A$
60 WEND
70
75 IF P$="XXX" THEN PRINT "ŢARA NECUNOSCUTĂ PROGRAMULUI"
80 DATA FRANŢA, PARIS
90 DATA ROMÂNIA, BUCUREŞTI
100 DATA BULGARIA, SOFIA
110 DATA SPANIA, MADRID
120 DATA ITALIA, ROMA
130 END
```

2. Să se determine lucrul mecanic total al cuplului motor de moment, pentru un arbore dat:

$$M_m=50\sqrt{2\,\theta^2+3\,\theta+4} \quad [\text{N}\cdot\text{m/rad}] \tag{14}$$

la o rotaţie $\theta \in [0,2\,\pi]$.

Indicaţie. Pentru rezolvarea problemei se utilizează relaţiile:

$$d\mathcal{L}=M\cdot\omega dt=Md\theta \tag{15}$$

Integrînd (14) obţinem:

$$\mathcal{L}=\int_{\theta_1}^{\theta_2} M\,d\theta \quad [\text{N}\cdot\text{m}] \tag{16}$$

Aşadar, problema se reduce la calculul integralei

$$\mathcal{L}m=\int_{\theta_1}^{\theta_2} M_m\,d\theta \tag{17}$$

Pentru calculul integralei (17) se va utiliza subrutina TRAPEZ. Se cunosc: θ1=30°; θ2=75°; număr intervale=10.

3. Introduceţi şi executaţi următorul program:

```
10 INPUT "NUMELE PRIETENULUI"; P$
20 WHILE N$< >"AAA" AND N$< >P$
30    READ N$, A$
40    IF N$=P$ THEN PRINT P$, A$
50 WEND
60 IF N$="xxx" THEN PRINT "NUME NECUNOSCUT PROGRAMULUI"
70 DATA IONESCU ION, 10 SEPTEMBRIE
80 DATA AGA ION, 5 IULIE
90 DATA LEU VASILE, 3 MAI
95 DATA xxx, xxx
100 END
```

4. Să se rezolve sistemul de ecuaţii (v. aplicaţie, pag. 319) prin metoda eliminării a lui GAUSS-JORDAN.

Tabela de variabile şi schema logică a programului (subrutinele 500; 5000) sînt prezentate în tabelul 6.13 respectiv figura 6.12.

Tabelul 6.13

TABELA DE VARIABILE

Variabile de intrare		Variabile de stare		Variabile de ieşire	
N1:	număr de linii	E1:	stare eroare	C1:	vector rezultate
N2:	număr de coloane	I2:	matrice de lucru		
A:	matricea coeficienţilor	M1:	lungime maximă		
Z:	vectorul termenilor liberi	B:	matrice de lucru		
		B1:	variabilă de lucru		
		D3:	determinant		
		H1:	variabilă de lucru		
		I3:	index linie		
		I4:	index coloană		
		I5:	variabilă de lucru		
		N3:	număr vectori coeficienţi		
		P1:	index pivot		
		W:	matricea soluţiilor		

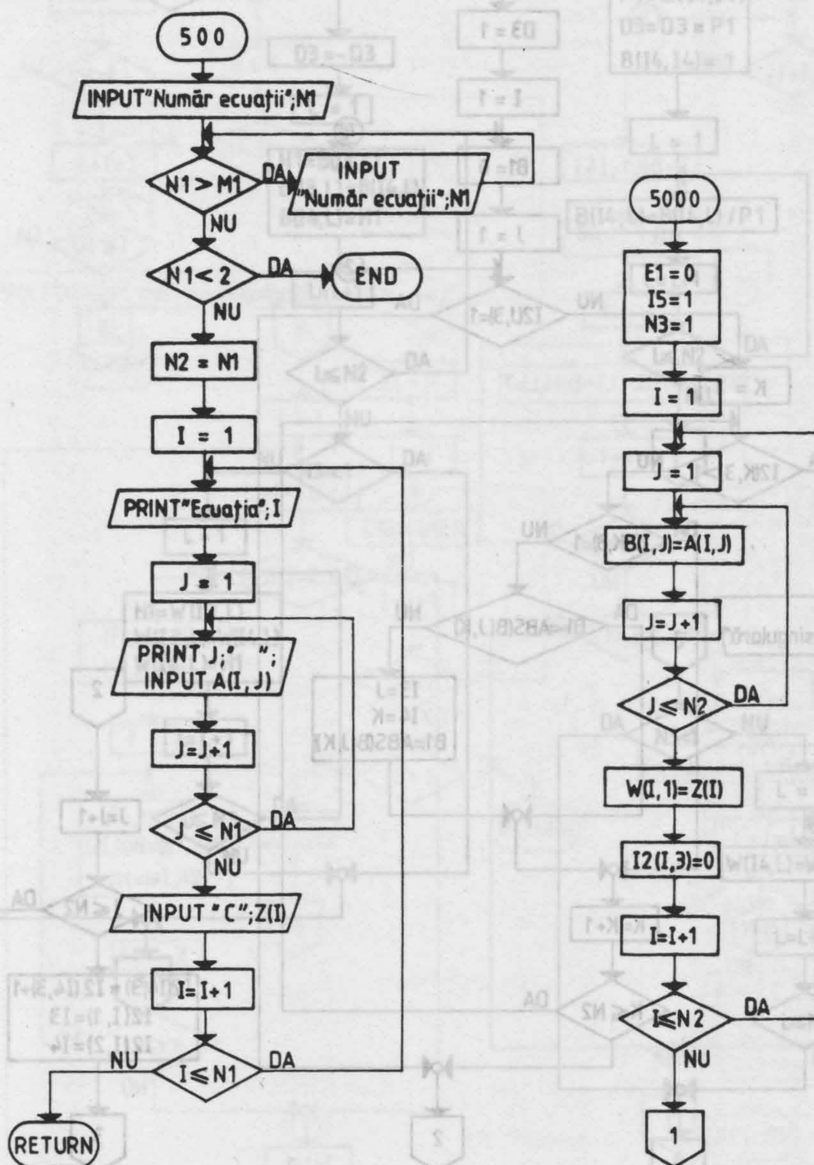

500
INPUT"Număr ecuaţii";N1
N1 > M1
DA
INPUT "Număr ecuaţii";N1
NU
N1 < 2
DA
END
NU
N2 = N1
I = 1
PRINT"Ecuaţia";I
J = 1
PRINT J;" ";
INPUT A(I, J)
J=J+1
J ≤ N1
DA
NU
INPUT " C";Z(I)
I=I+1
I ≤ N1
NU
DA
RETURN
5000
E1 = 0
I5 = 1
N3 = 1
I = 1
J = 1
B(I,J)=A(I,J)
J=J+1
J ≤ N2
DA
W(I, 1)= Z(I)
I2(I,3)=0
I=I+1
I ≤ N2
DA
NU
1

Fig. 6.12.

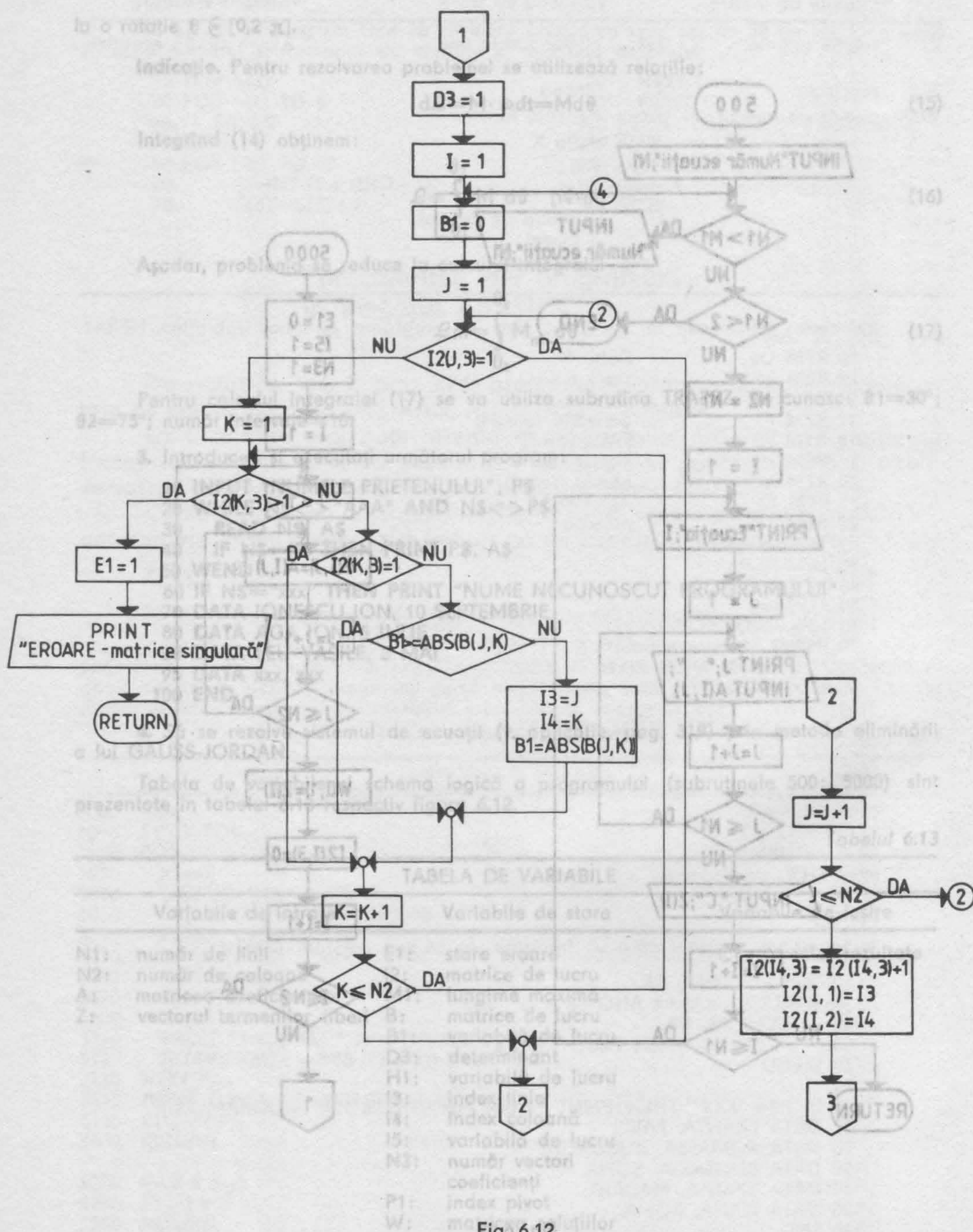

1
D3 = 1
I = 1
4
B1 = 0
J = 1
2
NU
I2(J,3)=1
DA
K = 1
DA
I2(K,3)>1
NU
E1=1
PRINT
"EROARE - matrice singulară"
RETURN
DA
I2(K,3)=1
NU
DA
B1=ABS(B(J,K)
NU
I3=J
I4=K
B1=ABS(B(J,K))
K=K+1
K≤N2
DA
2
2
J=J+1
J≤N2
DA
2
I2(I4,3) = I2(I4,3)+1
I2(I,1)=I3
I2(I,2)=I4
3

Fig. 6.12.

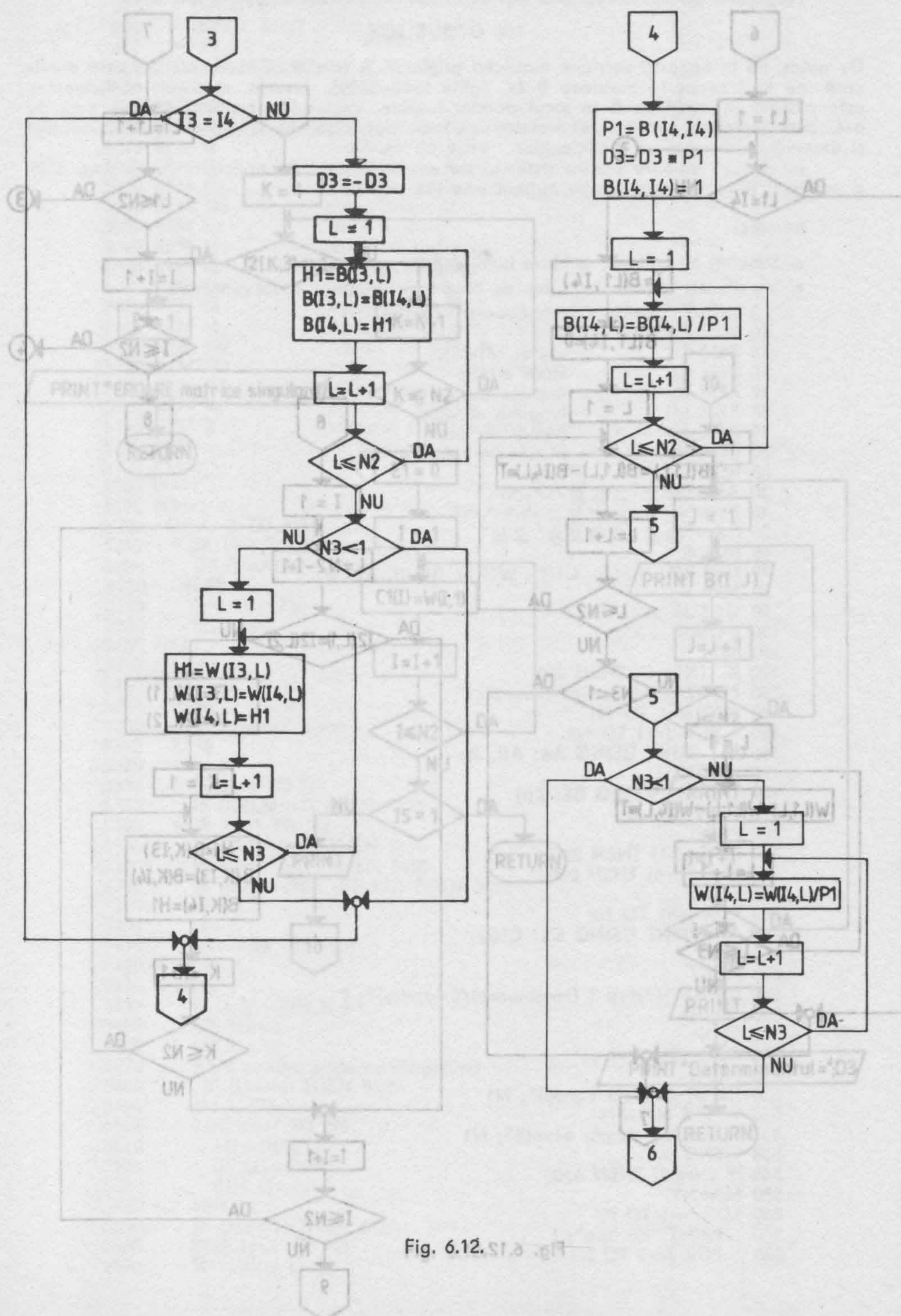

3
DA
I3 = I4
NU
D3=-D3
L = 1
H1=B(I3,L)
B(I3,L)=B(I4,L)
B(I4,L)=H1
L=L+1
L≤N2
DA
NU
NU
N3<1
DA
L = 1
H1=W(I3,L)
W(I3,L)=W(I4,L)
W(I4,L)=H1
L=L+1
L≤N3
DA
NU
4
4
P1=B(I4,I4)
D3=D3*P1
B(I4,I4)=1
L = 1
B(I4,L)=B(I4,L)/P1
L=L+1
L≤N2
DA
NU
5
5
DA
N3<1
NU
L = 1
W(I4,L)=W(I4,L)/P1
L=L+1
L≤N3
DA
NU
6

Fig. 6.12.

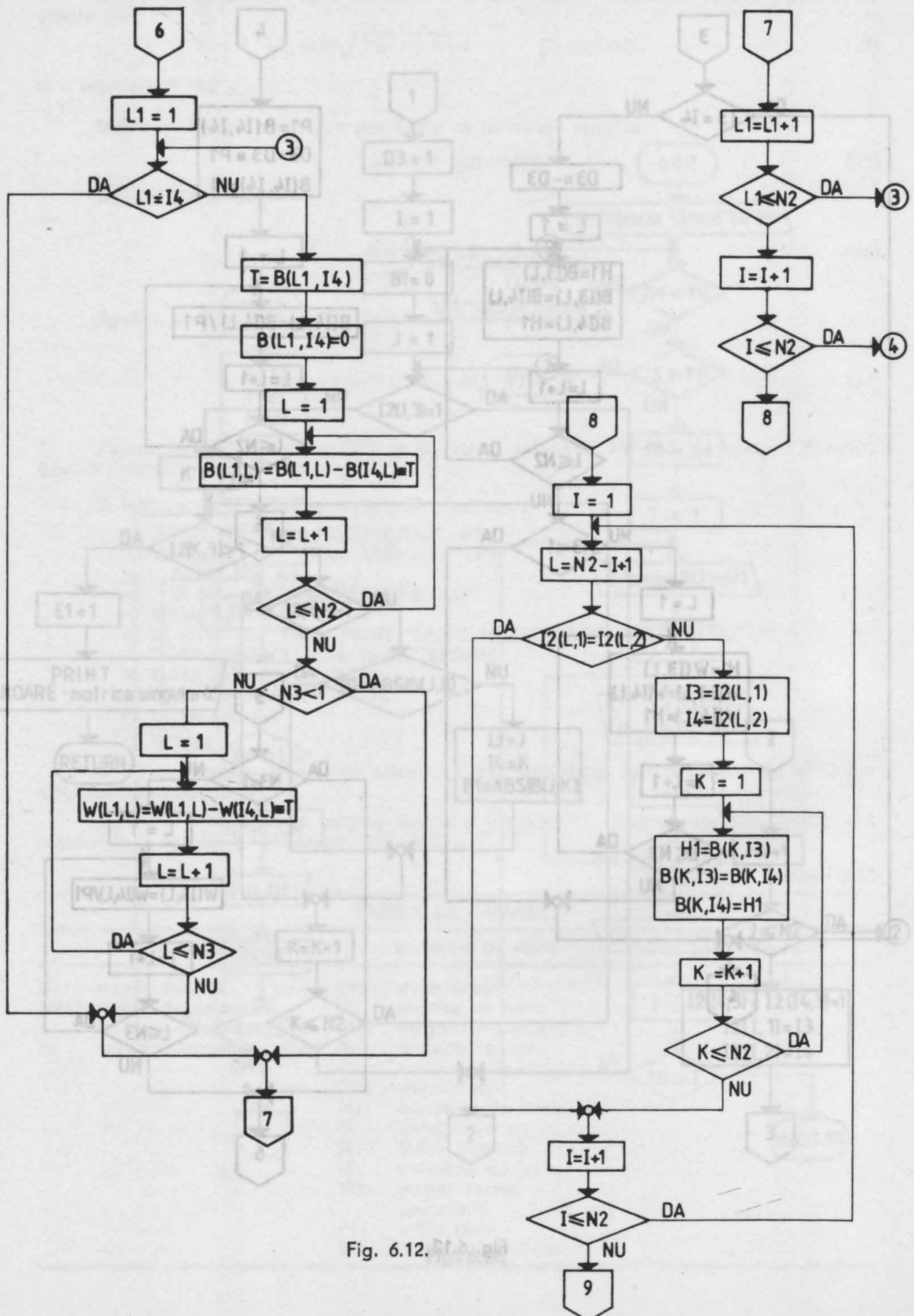
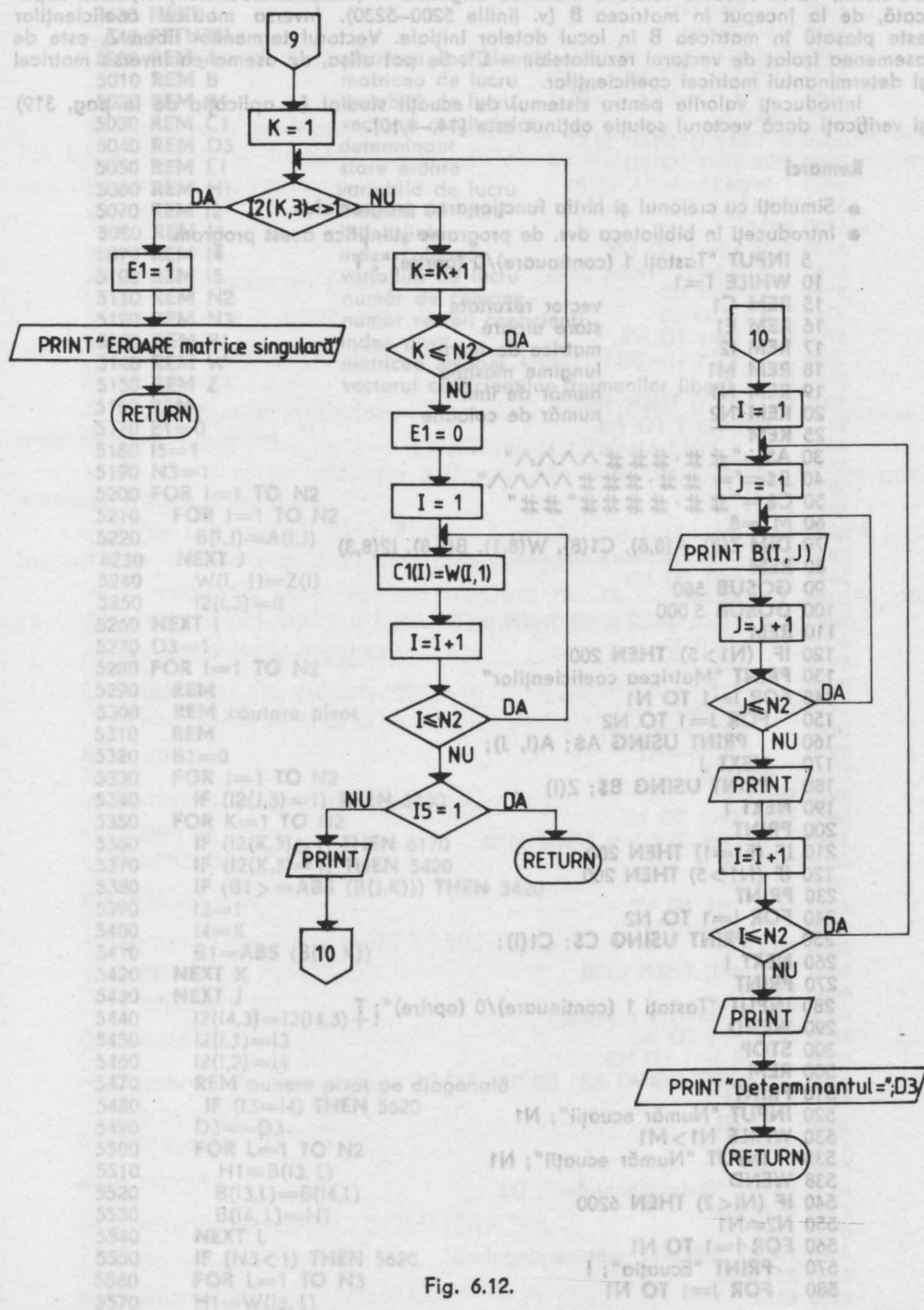

6
L1 = 1
3
DA
L1=I4
NU
T= B(L1 ,I4)
B(L1 ,I4)=0
L = 1
B(L1,L) = B(L1,L) − B(I4,L)*T
L= L+1
L≤N2
DA
NU
NU
N3<1
DA
L = 1
W(L1,L)=W(L1,L)−W(I4,L)*T
L= L+1
DA
L≤N3
NU
7
7
L1=L1+1
L1≤N2
DA
3
I=I+1
I≤N2
DA
4
8
8
I = 1
L=N2−I+1
DA
I2(L,1)=I2(L,2)
NU
I3=I2(L,1)
I4=I2(L,2)
K = 1
H1=B(K,I3)
B(K,I3)=B(K,I4)
B(K,I4)=H1
K =K+1
K≤N2
DA
NU
I=I+1
I≤N2
DA
NU
9

Fig. 6.12.



9
K = 1
DA
I2(K,3)<>1
NU
E1=1
PRINT "EROARE matrice singulară"
RETURN
K=K+1
K ≤ N2
DA
NU
E1 = 0
I = 1
C1(I)=W(I,1)
I=I+1
I≤N2
DA
NU
I5 = 1
NU
DA
PRINT
RETURN
10
10
I = 1
J = 1
PRINT B(I, J)
J=J +1
J≤N2
DA
NU
PRINT
I=I +1
I≤N2
DA
NU
PRINT
PRINT "Determinantul=";D3
RETURN

Fig. 6.12.

Programul principal apelează subrutina GAUSS-JORDAN prin instrucţiunea

100 **GOSUB** 5000

De notat, că în această versiune matricea originală A (matricea coeficienţilor) este duplicată, de la început în matricea B (v. liniile 5200–5230). Inversa matricei coeficienţilor este plasată în matricea B în locul datelor iniţiale. Vectorul termenilor liberi Z, este de asemenea izolat de vectorul rezultatelor – C1. Se pot afişa, de asemenea inversa matricei şi determinantul matricei coeficienţilor.

Introduceţi valorile pentru sistemul de ecuaţii studiat (v. aplicaţia de la pag. 319) şi verificaţi dacă vectorul soluţie obţinut este [14,–4,10].

Remarci

- Simulaţi cu creionul şi hîrtia funcţionarea programului.
- Introduceţi în biblioteca dvs. de programe ştiinţifice acest program.

```
  5 INPUT "Tastaţi 1 (continuare)/0 (oprire)"; T
 10 WHILE T=1
 15 REM C1          vector rezultate
 16 REM E1          stare eroare
 17 REM I2          matrice de lucru
 18 REM M1          lungime maximă
 19 REM N1          număr de linii
 20 REM N2          număr de coloane
 25 REM
 30 A$="##·###^^^^"
 40 B$="= ##·###^^^^"
 50 C$="##·####"##"
 60 M1=8
 70 DIM Z(8), A(8,8), C1(8), W(8,1), B(8,8), I2(8,3)
 80 REM
 90 GOSUB 500
100 GOSUB 5 000
110 REM
120 IF (N1>5) THEN 200
130 PRINT "Matricea coeficienţilor"
140 FOR I=1 TO N1
150   FOR J=1 TO N2
160     PRINT USING A$; A(I, J);
170   NEXT J
180   PRINT USING B$; Z(I)
190 NEXT I
200 PRINT
210 IF (E1=1) THEN 280
120 IF (N1>5) THEN 200
230 PRINT
240 FOR I=1 TO N2
250   PRINT USING C$; C1(I);
260 NEXT I
270 PRINT
280 INPUT "Tastaţi 1 (continuare)/0 (oprire)"; T
290 WEND
300 STOP
500 REM
510 PRINT
520 INPUT "Număr ecuaţii"; N1
530 WHILE N1>M1
535   INPUT "Număr ecuaţii"; N1
538 WEND
540 IF (NI<2) THEN 6200
550 N2=N1
560 FOR I=1 TO N1
570   PRINT "Ecuaţia"; I
580   FOR J=1 TO N1
```

```
590       PRINT J; "  ";
600       INPUT A(I,J)
610     NEXT J
620     INPUT "C"; Z(I)
630 NEXT I
640 RETURN
5000 REM A        matricea coeficienţilor
5010 REM B        matricea de lucru
5020 REM B1       variabilă de lucru
5030 REM C1       vectorul rezultatelor
5040 REM D3       determinant
5050 REM E1       stare eroare
5060 REM H1       variabila de lucru
5070 REM I2       matrice de lucru
5080 REM I3       index linie
5090 REM I4       index coloană
5100 REM I5       variabilă de lucru
5110 REM N2       număr de coloane
5120 REM N3       număr vectori coeficienţi
5130 REM P1       index pivot
5140 REM W        matricea soluţiei
5150 REM Z        vectorul coeficienţilor (termenilor liberi)
5160 REM
5170 E1=0
5180 I5=1
5190 N3=1
5200 FOR I=1 TO N2
5210   FOR J=1 TO N2
5220     B(I,J)=A(I,J)
4230   NEXT J
5240     W(I, 1)=Z(I)
5250     I2(I,3)=0
5260 NEXT I
5270 D3=1
5280 FOR I=1 TO N2
5290   REM
5300   REM căutare pivot
5310   REM
5320   B1=0
5330   FOR J=1 TO N2
5340   IF (I2(J,3)=1) THEN 5430
5350   FOR K=1 TO N2
5360   IF (I2(K,3)>1) THEN 6170
5370   IF (I2(K,3)=1) THEN 5420
5380   IF (B1>=ABS (B(J,K))) THEN 5420
5390     I3=J
5400     I4=K
5410     B1=ABS (B(J, K))
5420   NEXT K
5430   NEXT J
5440     I2(I4,3)=I2(I4,3)+1
5450     I2(I,1)=I3
5460     I2(I,2)=I4
5470     REM punere pivot pe diagonală
5480      IF (I3=I4) THEN 5620
5490     D3=-D3
5500     FOR L=1 TO N2
5510       H1=B(I3, L)
5520       B(I3,L)=B(I4,L)
5530       B(I4, L)=H1
5540     NEXT L
5550     IF (N3<1) THEN 5620
5560     FOR L=1 TO N3
5570     H1=W(I3, L)
```

```
5580   W(I3, L)=W(I4, L)
5590   W (I4, L)=H1
5600 NEXT L
5610 REM
5620 P1=B(I4, I4)
5630 D3=D3*P1
5640 B(I4, I4)=1
5650 FOR L=1 TO N2
5660   B(I4, L)=B(I4, L)/P1
5670 NEXT L
5680 IF (N3<1) THEN 5730
5690 FOR L=1 TO N3
5700   W(I4,L)=W(I4, L)/P1
5710 NEXT L
5720 REM
5730 FOR L1=1 TO N2
5740   IF (L1=I4) THEN 5835
5750   T=B(L1, I4)
5760   B(L1, I4)=0
5770 FOR L=1 TO N2
5780     B(L1, L)=B(L1, L)-B(I4, L) * T
5790 NEXT L
5800 IF (N3<1) THEN 5835
5810   FOR L=1 TO N3
5820           W(L1, L)=W(L1, L)-W(I4, L) * T
5830   NEXT L
5835 NEXT L1
5840 NEXT I
5850 REM
5860 REM interschimbare coloane
5870 FOR I=1 TO N2
5880   L=N2-I+1
5890   IF (I2 (L,1)=I2 (L,2)) THEN 5970
5900   I3=I2 (L,1)
5910   I4=I2 (L,2)
5920   FOR K=1 TO N2
5930     H1=B(K, I3)
5940     B(K,I3)=B(K,I4)
5950     B(K,I4)=H1
5960   NEXT K
5970 NEXT I
5980 FOR K=1 TO N2
5990   IF (I2(K, 3)<>1) THEN 6170
6000 NEXT K
6010 E1=0
6020 FOR I=1 TO N2
6030   C1(I)=W(I,1)
6040 NEXT I
6050 IF (I5=1) THEN 6190
6060 PRINT
6070 PRINT "Inversa matricei"
6080 FOR I=1 TO N2
6090   FOR J=1 TO N2
6100     PRINT USING A$; B(I,J);
6110   NEXT J
6120   PRINT
6130 NEXT I
6140 PRINT
6150 PRINT "Determinantul="; D3
6160 RETURN
6170 E1=1
6180 PRINT "EROARE matrice singulară"
6190 RETURN
6200 END
```

TEMA 6

☐ Răspundeţi prin DA sau NU la următoarele întrebări:

Instrucţiunile matriciale sînt precedate de cuvîntul cheie **MAT.** Limbajul BASIC-aMIC şi limbajul BASIC-PLUS permit efectuarea următoarelor operaţii cu masive, luate ca un tot:
- adunarea,
- scăderea,
- înmulţirea cu un scalar,
- împărţirea a două masive,
- înmulţirea a două masive,
- atribuirea.

– Instrucţiunile matriciale **ZER, COD, IDN** se folosesc la iniţializarea unor masive numerice.

– Atunci cînd se foloseşte instrucţiunea **MAT INPUT** elementele unui masiv se introduc pe linie.

– Unei instrucţiuni **MAT READ** i se asociază instrucţiunea **DATA** (corespunzător elementelor din liniile masivului).

– În cazul în care într-un program nu se lucrează cu masive ale căror dimensiuni sînt mai mari ca 10 instrucţiunea **DIM** poate fi omisă.

– Secvenţa de instrucţiuni:

```
50 DIM A(11,11), B(11,11), C(11,11), D(11,11)
60 MAT INPUT A, B, C, D
```

este corectă.

– Secvenţa de instrucţiuni BASIC-aMIC:

```
10 DIM A$ (20, 5)
20 INPUT A$
```

nu este corectă.

– Secvenţa de instrucţiuni BASIC-PLUS

```
10 DIM A(6,9)
15 MAT=INV (A)
```

nu este corectă.

– Secvenţa de instrucţiuni BASIC-PLUS

```
10 DIM A(6,6)
15 MAT=INV (A)
```

este corectă.

– Instrucţiunea BASIC-COMMODORE

```
10 A=A+B(I, -2)
```

este incorectă.

– Secvenţa de instrucţiuni BASIC-PLUS

```
10 DIM A(30)
20 MAT A=IDN
```

este incorectă.

– Instrucţiunea

```
10 DIM A(20), B(10,15), X, Y, Z, U(15)
```

este eronată.

– Instrucţiunea ABASIC

```
10 X(A(I))=Y(A(I+1))+3
```

este corectă.

– Următorul program:

```
10 DIM B(4,5)
20 FOR L=1 TO 4
30 FOR C=1 TO 5
40 B (L, C)=8
50 NEXT C
60 NEXT L
70 FOR L=1 TO 4
80 FOR C=1 TO 5
90 PRINT B (L,C);
100 NEXT C
110 PRINT
120 PRINT
130 NEXT L
140 END
```

afişează cifra 8 de 20 de ori, într-o matrice cu 4 linii şi 5 coloane.
– Funcţia **RANDOMIZE** iniţializează generatorul de numere aleatoare.
– În urma execuţiei programului

```
10 FOR I=1 TO 3
20   PRINT RND
30 NEXT I
```

se va genera aceeaşi secvenţă de numere aleatoare.
– În BASIC-aMIC punctul de coordonate (1,1) se află în colţul din stînga sus al ecranului iar punctul de coordonate (32, 32) în dreapta jos.
– În BASIC HC-85 variabilele şir şi variabilele indexate pot avea acelaşi nume.
– Variabilele şir trebuie declarate într-o instrucţiune **DIM**.

☐ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Operaţia de adunare a masivului A cu masivul X, cu plasarea rezultatului în masivul Y, se realizează cu instrucţiunea matricială ....

b) Operaţia de scădere a două masive X şi Y, cu plasarea rezultatului în masivul Z se realizează cu instrucţiunea matricială ....

c) Operaţia de înmulţire a matricii X cu scalarul S se realizează cu instrucţiunea matricială ....

d) Operaţia de înmulţire a două masive X şi Y, cu plasarea rezultatului în masivul Z se realizează cu instrucţiunea matricială ....

e) Instrucţiunile matriciale prezintă următoarele avantaje..........

f) Deosebirea între o instrucţiune **MAT READ** şi o instrucţiune **MAT INPUT** constă în ....

g) Secvenţa de instrucţiuni:

```
10 INPUT C, L
20 DIM A(C,L), B(C), X(L)
```

este corectă numai dacă ....

h) Pentru rezolvarea unei ecuaţii prin metoda Newton-Raphson se introduc următoarele date ....

i) Pentru calculul integralelor prin metoda trapezelor trebuie cunoscute ....

j) În urma execuţiei programului:

```
10 DIM X(6,6)
15 MAT READ X (4,5)
20 MAT PRINT X;
30 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
40 DATA 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
50 DATA 21, 22, 23, 24, 25, 26, 27, 8, 29, 30
60 DATA 31, 32, 33, 34, 35, 36
70 END
```

rezultă ....

k) În urma execuţiei programului:

```
10 DIM A(1,4)                 40 DATA 1, 2, 3, 4
20 MAT READ A                 50 END
30 MAT PRINT A
```

rezultă ....

l) În urma execuţiei programului:

```
10 DIM Y(4,1)                 40 DATA 1, 2, 3, 4
20 MAT READ A                 50 END
30 MAT PRINT A
```

rezultă ....

m) în urma execuţiei programului:

```
10 DIM B(4,5)                 40 DATA 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
20 MAT READ B                 50 DATA 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
30 MAT PRINT B;               60 END
```

se afişează 4 linii cu .... şi .... coloane cu ....

n) În BASIC-aMIC, într-o instrucţiune **PRINT AT** coordonatele pentru ⟨nr. linie⟩ şi ⟨nr. coloane⟩ se transmit .... iar în BASIC-PRAE ....

o) În BASIC-PRAE simbolurile folosite pentru editarea valorilor numerice sînt ....

p) În instrucţiunea **PRINT** din limbajul BASIC TIM S se pot specifica linia .... şi coloana ....

r) Limbajul BASIC implementat pe calculatoarele .... nu acceptă instrucţiunile matriciale.

s) Instrucţiunea **PRINT AT** este acceptată în BASIC ....

t) În BASIC ...., ... este posibilă utilizarea unor subşiruri dintr-un şir.

u) Ecranul TV, din punctul de vedere al PRAE-ului este împărţit în .... puncte pe orizontală şi .... puncte pe verticală. Punctul de coordonare (0, 0) este stabilit în ....

☐ Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Se consideră matricile

$$A=\begin{bmatrix}1 & -2 & 3\\ 5 & -1 & -2\\ 0 & 3 & 4\end{bmatrix};\quad B=\begin{bmatrix}2 & -4 & 0\\ -3 & 1 & 2\\ 5 & 2 & -5\end{bmatrix}$$

Să se calculeze:

a1) $(A*B)^{-1}$

a2) $B^{-1}*A^{-1}$

b) Se consideră matricea:

$$Y=\begin{bmatrix}4 & -4 & 4\\ 1 & 1 & 7\\ -3 & 9 & -8\end{bmatrix}$$

Să se calculeze:

b1) Y^{-1}

b2) $Y*Y^{-1}$

b3) $Y^{-1}*Y$

c) Să se calculeze:

$$\left[[1\quad 1\quad 1]\cdot\begin{bmatrix}20 & 700 & 30\\ 40 & 21 & 10\\ 0 & 400 & 15\end{bmatrix}\right]\cdot\begin{bmatrix}2.05\\ .23\\ .30\\ .25\end{bmatrix}$$

d) Să se citească matricea

$$\begin{bmatrix}1 & 6 & 11\\ 2 & 7 & 12\\ 3 & 8 & 13\\ 4 & 9 & 14\\ 5 & 10 & 15\end{bmatrix}$$

şi să se tipărească sub forma:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

e) Să se rezolve următoarele sisteme de ecuaţii:

e1)
$$-3w+6x-5y-z=-32$$
$$w+9x-5y-2z=9$$
$$w+6y+5z=2$$
$$-7w+4x-y+5z=-56$$

e2)
$$7v+6w-3x-y+9z=26.3$$
$$-9v+2w+9x+5y+z=91.1$$
$$-3v+4w+5x+5z=62.9$$
$$6v-8x-2y-6z=-55.6$$
$$-3v-9w-5x+7y+3z=-25.9$$

e3)
$$-2x-5y=-16$$
$$-x+4y=31$$

f) Pentru termocuplul fier-constantan, dependenţele dintre tensiunea electromotoare şi agitaţia termică măsurată prin temperatură pot fi aproximate prin drepte ai căror coeficienţi se pot determina printr-o regresie liniară.

În tabelul 6.14 sînt ilustrate rezultatele măsurătorilor de laborator.

Tabelul 6.14

T°C	(mv)	T°C	(mv)
24	0,11	40	1,06
26	0,22	41	1,1
28	0,38	44	1,24
29	0,46	48	1,48
32	0,58	50	1,56
33	0,69	53	1,79
35	0,80	58	2,02

La proiectarea şi elaborarea programului se vor avea în vedere următoarele funcţiuni:

1. Citirea datelor de intrare
2. Afişarea datelor de intrare
3. Apelarea subrutinei de regresie liniară REGLIN (500).
4. Imprimarea coeficienţilor (a, b) dreptei de regresie.

În tabelul 6.15 şi tabelul 6.16 se prezintă tabela de variabile a subrutinei respectiv programul sursă al aceleaşi subrutine.

Tabelul 6.15

TABELA DE VARIABILE		
Variabile de intrare	Variabile de stare	Variabile de ieşire
N: numărul valorilor experimentale	S1, S2, S3, S4: variabile de lucru	A: termenul liber
X: vectorul valorilor variabilei independente		B: panta dreptei de regresie
Y: vectorul valorilor variabilei dependente		

Tabelul 6.16

PROGRAM SURSA
500 S1=0.0
510 S2=0.0
520 S3=0.0
530 S4=0.0
550 **FOR** I=1 **TO** N
560 S1=S1+X(I)
570 S2=S2+Y(I)
580 S3=S3+X(I) * X(I)
590 S4=S4+X(I) * Y(I)
600 **NEXT** I
610 A=(S2 * S3–S4 * S1)/(N * S3–S1 * S1)
620 B=(N * S4–S1 * S2)/(N * S3–S1 * S1)
630 **RETURN**

Observaţie. Pentru calculul coeficienţilor (a, b) ecuaţiei de regresie

$$y=ax+b \tag{18}$$

s-au folosit formulele:

$$a=\frac{\Sigma y_i \Sigma x_i^2-\Sigma x_i \Sigma x_i y_i}{n\Sigma x_i^2-(\Sigma x_i)^2} \tag{19}$$

$$b=\frac{n\Sigma x_i y_i-\Sigma x_i \Sigma y_i}{n\Sigma x_i^2-(\Sigma x_i)^2} \tag{20}$$

g) Pentru calculul geometric al profilului flancurilor dintelui, în cazul angrenajelor cu dinţi drepţi şi profil în evolventă se utilizează funcţia involută

$$inv(\alpha)=tg\alpha-\alpha \tag{21}$$

Utilizînd metoda Newton-Raphson, să se determine valoarea unghiului α din ecuaţia:

$$tg\alpha-\alpha=\gamma \tag{22}$$

pentru: $\alpha_1=20°$, precizia=0,000001, numărul de iteraţii=10, $\gamma=15°$. Se va apela subrutina NERA1 (v. pag. 342)

h) Să se determine prin metoda Newton-Raphson soluţia ecuaţiei

$$e^x=4x$$

cunoscînd: soluţia iniţială x=0,1, numărul de iteraţii=20 şi precizia==0,000001. Se va apela subrutina NERA1.

Indicaţie. Pentru evaluarea lui e^x folosiţi secvenţa de instrucţiuni:

```
5200 E=EXP (X)
5210 F=E–4*X
5220 F1=E–4
5230 RETURN
```

i) În tabelul 6.17 se prezintă datele experimentale privind variaţia unei rezistenţe cu temperatura în domeniul T∈[50, 100°C].

Tabelul 6.17

T [°C]	50	55	60	65	70	75	80	85	90	95	100
R [Ω]	76	76,11	76,32	76,47	76,50	76,90	77,1	77,32	77,31	77,42	77,56

Să se determine valorile intermediare ale rezistenţei conductorului prin **interpolare liniară.**

La proiectarea şi elaborarea programului se vor avea în vedere următoarele funcţiuni:

1. Citirea/scrierea datelor de intrare
2. Apelarea subrutinei INTERLIN
3. Tipărire rezultate.

În tabelul 6.18, tabelul 6.19 şi tabelul 6.20 se prezintă formatul datelor de ieşire, tabela de variabile a subrutinei INTERLIN respectiv programul sursă al aceleiaşi subrutine.

Tabelul 6.18

FORMATUL DATELOR DE IEŞIRE

DATELE DE INTRARE

T (GR.C)

** · *
** · *
.
.
.
** · *

R (OHMI)

** · *
** · *
.
.
.
** · *

DATELE DE IEŞIRE

T= ** · * GR·C R= ** · ** OHMI

Tabelul 6.19

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieşire
N: Numărul perechilor de valori cunoscute din graficul funcţiei X: vectorul valorilor argumentului Y: vectorul valorilor funcţiei X0: valoarea iniţială a argumentului	KOD: cod de eroare (0 – interpolare reuşită; 1 – eroare) U, V: variabile de lucru	Y0: valoarea aproximativă a funcţiei pentru valoarea X0 dată KOD: cod eroare

Tabelul 6.20

PROGRAM SURSĂ
500 KOD=0
510 N1=N-1
520 FOR I=1 TO N1
530 IF (X0>=X(I)) AND (X0<=X(I+1)) THEN 570
540 NEXT I
550 KOD=1
560 RETURN
570 U=Y(I)+(Y(I+1)-Y(I))
580 V=X(I+1)-X(I)) * (X0-X(I))
590 Y0=U/V
600 RETURN

j) Să se determine intensităţile curenţilor din laturile circuitului din figura 6.13 prin metoda teoremelor lui Kirchoff. Se dau:

$$E1=8V;\ E2=48V;\ r1=3\Omega;\ r2=2\Omega;\ R=2\Omega.$$

Pentru rezolvarea sistemului de ecuaţii se vor utiliza:

- formulele lui Cramer (v. pag. 319),
- metoda eliminării a lui Gauss (v. subrutinele 500, 5000, pag. 331),
- metoda eliminării a lui Gauss-Jordan (v. subrutinele 500, 5000, pag. 344).

k) Scrieţi un program care să simuleze aruncarea a două monezi de 30 de ori. Programul va contoriza şi afişa numărul pentru care rezultatul acestui experiment imaginar este: leu–leu (LL), stemă–stemă (SS) şi leu–stemă (LS) sau stemă–leu (SL).

l) Scrieţi un program care să furnizeze, la cerere, informaţii privind autorii cărţilor din biblioteca dvs. personală (nume, pronume, lucrare, editură, anul apariţiei).

m) Se consideră trei rezervoare cilindrice echilaterale care conţin benzină de trei calităţi. Să se editeze situaţia livrărilor de benzină pentru un număr oarecare (N) de beneficiari. Programul va afişa în final "TOTAL LIVRĂRI" şi "TOTAL VALOARE". Preţurile de livrare pe tona de benzină sînt respectiv 8 750 lei, 7 500 lei şi 5 250 lei. Formatul datelor de ieşire, tabela de variabile, pseudocodul şi textul conversaţiei sînt prezentate în modulele de analiză şi proiectare structurată figura 6.14.

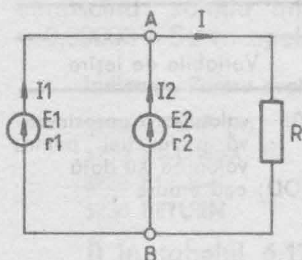


Fig. 6.13.

☐ **Se reia problema din conversaţia** precedentă, complicînd-o după cum urmează.

- Se citeşte de la tastatură numărul de articole (maximum 8);
- Se memorează într-o matrice cu 8 linii (reprezentînd numărul maxim de articole – tipuri de jucării) şi 12 coloane (reprezentînd lunile anului) cantităţile lunare de jucării fabricate.

FORMATUL DATELOR DE IEŞIRE

* * SITUAŢIA LIVRĂRILOR * *

NR.	DENUMIRE	LIVRĂRI	VALOARE
*	* * * * *	* * *	* * *
.	.	.	.
.	.	.	.
.	.	.	.

TOTAL LIVRĂRI: * * TONE

TOTAL VALOARE: * * LEI

a)

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieşire
B$: vector nume beneficiari	S: total livrări pe beneficiar	S1: total livrări
L: matrice livrări benzină	T: total valoare livrări pe beneficiar	T1: total valoare livrări
N: număr beneficiari	I: index	S: total livrări pe beneficiar
P: vector preţuri pe tona de benzină	J: index	T: total valoare livrări pe beneficiar

b)

PSEUDOCODUL

```
TEMA          SEQ
                  INPUT "INTRODUCEŢI NUMĂR BENEFICIARI"; N
                  PRINT
CIT-PRET      PENTRU I DE LA 1 LA 3
                      READ P(I)
CIT-PRET      SFÎRŞIT
                  PRINT
CIT-NUM-LIV   PENTRU I DE LA 1 LA N
                  INPUT "NUME BENEFICIAR"; B$(I)
CIT-LIV           PENTRU J DE LA 1 LA 3
                      PRINT "LIVRĂRI"; B$(I); "DIN REZERVORUL"; J
                      INPUT L (I, J)
CIT-LIV           SFÎRŞIT
CIT-NUM-LIV   SFÎRŞIT
A             SEQ
                  PRINT "SITUAŢIA LIVRĂRILOR"
                  PRINT
                  PRINT "NR. DENUMIRE LIVRĂRI VALOARE"
                  PRINT
```

c)

Fig. 6.14. Modulele de analiză şi proiectare structurată: *a)* formatul datelor de ieşire; *b)* tabela de variabile; *c)* pseudocodul;

```
              PRINT
              PRINT
              PRINT "— — — — — — — — — — — — — —"
              T1=0
              S1=0
A             END
TOTAL         PENTRU I DE LA 1 LA N
              S=0
              T=0
T1            PENTRU J DE LA 1 LA 3
T2              DACĂ L (I, J)< >0
                  T=T+L(I, J) * P(J)
                  S=S+L (I, J)
T2              SFÎRŞIT
T1            SFÎRŞIT
C             SEQ
              S1=S1+S
              T1=T1+T
              PRINT
C             END
              PRINT I; B$(I); S; T
TOTAL         SFÎRŞIT
F             SEQ
              PRINT "— — — — — — — — — — — — — —"
              PRINT "TOTAL LIVRĂRI : " ; S1 ; "TONE"
              PRINT " — — — — — — — — — — — —"
              PRINT "TOTAL VALOARE : " ; T1 ; "LEI"
F             END
TEMA          END
```

c)

TEXTUL CONVERSAŢIEI

```
RUN
INTRODUCEŢI NUMĂR BENEFICIARI   ?  5
NUME BENEFICIAR? PECO-11
LIVRĂRI PECO-11 DIN REZERVORUL  1  ?  1
LIVRĂRI PECO-11 DIN REZERVORUL  2  ?  12
LIVRĂRI PECO-11 DIN REZERVORUL  3  ?  0
NUME BENEFICIAR? PECO-12
LIVRĂRI PECO-12 DIN REZERVORUL  1  ?  5
LIVRĂRI PECO-12 DIN REZERVORUL  2  ?  3
LIVRĂRI PECO-12 DIN REZERVORUL  3  ?  12
NUME BENEFICIAR? PECO-13
LIVRĂRI PECO-13 DIN REZERVORUL  1  ?  4
LIVRĂRI PECO-13 DIN REZERVORUL  2  ?  5
LIVRĂRI PECO-13 DIN REZERVORUL  3  ?  0
NUME BENEFICIAR? PECO-14
LIVRĂRI PECO-14 DIN REZERVORUL  1  ?  2
LIVRĂRI PECO-14 DIN REZERVORUL  2  ?  3
LIVRĂRI PECO-14 DIN REZERVORUL  3  ?  5
NUME BENEFICIAR? PECO-15
LIVRĂRI PECO-15 DIN REZERVORUL  1  ?  2
LIVRĂRI PECO-15 DIN REZERVORUL  2  ?  0
LIVRĂRI PECO-15 DIN REZERVORUL  3  ?  5
```

d)

Fig. 6.14. *d)* textul conversaţiei.

• Pentru fiecare tip de jucărie fabricată, în cadrul fiecărui trimestru se calculează:

– procentul cantităţii totale trimestriale faţă de cantitatea anuală;

– abaterea cantităţii totale trimestriale (în procente) faţă de cantitatea medie trimestrială.

• Pentru fiecare articol se afişează:

– număr trimestru (I, II, III, IV);

– cantitatea totală trimestrială;

– procentul cantităţii trimestriale faţă de cantitatea anuală;

– abaterea cantităţii totale trimestriale (în procente) faţă de cantitatea medie trimestrială.

☐ **Se reia problema din conversaţia precedentă complicînd-o sub următoarele aspecte:**

• Se citeşte de la tastatură numărul de articole.

• Se memorează codurile tuturor articolelor.

• Pentru fiecare articol se memorează consumurile trimestriale normate.

• Într-o matrice cu patru linii (reprezentînd trimestrele) şi trei coloane (reprezentînd lunile unui trimestru) se memorează consumurile trimestriale raportate ale unui articol identificat prin codul său.

• Pentru fiecare articol se va calcula abaterea faţă de consumurile trimestriale normate, rezultatele urmînd a se memora într-o matrice cu 4 coloane reprezentînd abaterile trimestriale şi n linii reprezentînd numărul articolelor (maximum 20).

Datele vor fi introduse dinamic iar rezultatele vor fi tipărite în formatul: COD, TRIMESTRUL 1, TRIMESTRUL 2, TRIMESTRUL 3, TRIMESTRUL 4.

SOLUŢIA TEMEI 6

☐ Nu răspundem.

☐ Nu răspundem.

☐ **Programele BASIC sînt:**

a)
```
10 DIM A(3,3), B(3,3), C(3,3), D(3,3), E(3,3)
20 MAT READ A, B
30 MAT C=A*B
40 MAT D=INV (C)
50 PRINT
60 PRINT "INV (A*B)"
70 MAT PRINT D
80 MAT C=INV (A)
90 MAT D=INV (B)
100 MAT E=D*C
110 PRINT
120 PRINT "INV (B)*INV (A)"
130 MAT PRINT E
```

```
140 DATA 1, -2, 3, 5, -1, -2, 0, 3, 4
150 DATA 2, -4, 0, -3, 1, 2, 5, 2, -5
160 END
```

b)

```
10 DIM A(3,3), B(3,3), C(3,3)
20 MAT READ A
30 MAT B=INV (A)
40 PRINT "INV (A)"
50 MAT PRINT B
60 MAT C=A*B
70 PRINT "A*INV (A)"
90 MAT PRINT C
100 MAT C=B*A
110 PRINT
120 PRINT "INV (A)*A"
130 MAT PRINT C
140 DATA 4, -4, 4, 1, 1, 7, -3, 9, -8
150 END
```

c)

```
10 DIM A(1,3), B(3,4), D(1,4), E(1,4), C(4,1)
20 MAT READ A, B, C
30 MAT D=A*B
40 MAT E=D*C
50 MAT PRINT E
60 DATA 1, 1, 1
70 DATA 20, 700, 30, 40, 21, 10, 0, 400, 15
80 DATA 2.05, .23, .30, .25
90 END
```

e)

```
10 DIM C(10,10), K(10,1), I(10,10), S(10,1)
20 READ N
30 IF N=0 THEN 420
40 MAT READ C(N,N), K(N,1)
50 PRINT "MATRICEA COEFICIENŢILOR"
60 MAT PRINT C
70 PRINT
80 PRINT "TERMENII LIBERI"
90 MAT PRINT K
100 MAT I=INV (C)
110 MAT S=I*K
120 PRINT
130 PRINT "SOLUŢIE"
140 MAT PRINT S
240 PRINT
250 GO TO 20
260 DATA 4
270 DATA -3, 6, -5, -1
280 DATA 1, 9, -5, -2
290 DATA 1, 0, 6, 5, -7, 4, -1, 5, -32, 9, 2, -56
310 DATA 5
320 DATA 7, 6, -3, -1, 9
330 DATA -9, 2, 9, 5, 1, -3, 4, 5, 5
340 DATA 6, 0, -8, -2, -6
350 DATA -3, -9, 5, 7, 3
360 DATA 26.3, 91.1, 62.9, -55.6, -25.9
370 DATA 2
380 DATA -2, -5
390 DATA -1, 4
400 DATA -16, 31, 0
420 END
```

k)

```
5 REM LL : LEU-LEU; SS : STEMA-STEMA; LS : LEU-STEMA SAU
6 REM STEMA-LEU
10 LL=0
20 SS=0
```

```
30 LS=0
40 FOR I=1 TO 30
50 F1=INT (2*RND+1)
60 F2=INT (2*RND+1)
70 S=F1+F2
80 IF S=4 THEN
90   LL=LL+1
100 ELSE
110 IF S=2 THEN
120 SS=SS+1
130 ELSE
140 LS=LS+1
150 ENDIF
160 ENDIF
170 PRINT "SS"; TAB 10; "LT"; TAB(20); "SS"
180 PRINT SS; TAB 10; LT; TAB(20); SS
190 END
```

l)

```
10 INPUT "TITLUL LUCRĂRII"; N$
40 WHILE N$< >A$ AND N$< >"xxx"
50   READ A$, B$, C$, D$, E$
60   IF N$=A$ THEN PRINT A$, B$, C$, D$, E$
70 WEND
80 DATA REBREANU, LIVIU, ION, EMINESCU, 1980
90 DATA REBREANU, LIVIU, RĂSCOALA, EMINESCU, 1980
100 DATA REBREANU, LIVIU, CIULEANDRA, EMINESCU 1981
110 DATA xxx, xxx, xxx, xxx, xxx
120 IF N$="xxx" THEN PRINT "AUTOR NECUNOSCUT"
130 END
```

m)

```
100 REM PROGRAM DE EDITARE A SITUAŢIILOR LIVRĂRILOR
110 REM DE BENZINĂ PRIVIND UN NUMĂR OARECARE (N)
120 REM DE BENEFICIARI
130 PRINT
140 DIM P(3), B$(20), L(20, 3)
150 REM P – PREŢURI PE TONĂ DE BENZINĂ
160 REM B$ – NUME BENEFICIARI
170 REM L – LIVRĂRI PENTRU FIECARE BENEFICIAR
180 REM N – NUMĂR DE BENEFICIARI
190 REM CITIRE NUMĂR BENEFICIARI
200 INPUT "INTRODUCEŢI NUMĂR BENEFICIARI"; N
210 PRINT
230 DATA 8750, 7500, 5250
240 FOR I=1 TO 3
250   READ P(I)
260 NEXT I
270 PRINT
280 REM CITIRE NUME ŞI LIVRĂRI BENEFICIAR
290 FOR I=1 TO N
300 INPUT "NUME BENEFICIAR"; B$(I)
310 FOR J=1 TO 3
320 PRINT "LIVRĂRI"; B$(I); "DIN REZERVORUL"; J
330 INPUT L(I, J)
340 NEXT J
350 NEXT I
360 REM TIPĂRIRE RAPORT LIVRĂRI
370 PRINT TAB(6), "**SITUAŢIA LIVRĂRILOR**"
380 PRINT
390 PRINT TAB(2); "NR"; TAB(5); "DENUMIRE"; TAB(21); "LIVRĂRI"; TAB(30);
400 PRINT "VALOARE"
410 PRINT
```

```
420 GOSUB 620
430 T1=0 : S1=0
440 FOR I=1 TO N
450 S=0 : T=0
460 FOR J=1 TO 3
470   IF L(I, J)=0 THEN 500
480   T=T+L(I, J)*P(J)
490   S=S+L(I, J)
500 NEXT J
510 S1=S1+S
520 T1=T1+T
530 PRINT TAB(1); I; TAB(6); B$(I); TAB(24); S; TAB(30); T
540 NEXT I
550 GOSUB 620
560 PRINT "        TOTAL LIVRĂRI : "; S1; "TONE"
570 GOSUB 620
580 PRINT "        TOTAL VALOARE : "; T1; "LEI"
590 PRINT FOR I=1 TO 3
600 STOP
610 END
620 PRINT "-------------"
630 RETURN
```

```
:RUN

 INTRODUCETI NUMAR BENEFICIARI ?5


 NUME BENEFICIAR ?PECO-1
 LIVRARI PECO-1 DIN REZERVORUL  1
?100
 LIVRARI PECO-1 DIN REZERVORUL  2
?20
 LIVRARI PECO-1 DIN REZERVORUL  3
?30
 NUME BENEFICIAR ?PECO-2
 LIVRARI PECO-2 DIN REZERVORUL  1
?10
 LIVRARI PECO-2 DIN REZERVORUL  2
?20
 LIVRARI PECO-2 DIN REZERVORUL  3
?30
 NUME BENEFICIAR ?PECO-3
 LIVRARI PECO-3 DIN REZERVORUL  1
?10
 LIVRARI PECO-3 DIN REZERVORUL  2
?11
 LIVRARI PECO-3 DIN REZERVORUL  3
?12
 NUME BENEFICIAR ?PECO-4
 LIVRARI PECO-4 DIN REZERVORUL  1
?13
 LIVRARI PECO-4 DIN REZERVORUL  2
?14
 LIVRARI PECO-4 DIN REZERVORUL  3
?15
 NUME BENEFICIAR ?PECO-5
 LIVRARI PECO-5 DIN REZERVORUL  1
?23
```

```
LIVRARI PECO-5 DIN REZERVORUL 2
?7
LIVRARI PECO-5 DIN REZERVORUL 3
?0
              ** SITUATIA LIVRARILOR **

NR DENUMIRE         LIVRARI  VALOARE
-----------------------------------------
1  PECO-1            150    1.1825E+06
2  PECO-2            60     395000
3  PECO-3            33     233000
4  PECO-4            42     297500
5  PECO-5            30     253750
-----------------------------------------
        TOTAL LIVRARI :  315  TONE
-----------------------------------------
        TOTAL VALOARE :  2.36175E+06  LEI

STOP AT LINE 600
```

Programul pentru calculul abaterii cantității totale trimestriale de jucării, în procente față de cantitatea medie trimestrială este:

```
5 REM PROGRAMUL CITEȘTE DE LA TASTATURĂ NUMĂRUL DE
10 REM ARTICOLE (Z) ȘI CANTITĂȚILE LUNARE DE JUCĂRII
15 REM FABRICATE (C(K, I)).
20 REM PROGRAMUL CALCULEAZĂ ȘI AFIȘEAZĂ PROCENTUL CANTITĂȚII
25 REM TRIMESTRIALE (R1) FAȚĂ DE CANTITATEA
30 REM ANUALĂ (T)
35 REM PROGRAMUL CALCULEAZĂ ȘI AFIȘEAZĂ ABATEREA CANTITĂȚII
40 REM TOTALE TRIMESTRIALE (R2) ÎN PROCENTE
45 REM FAȚĂ DE CANTITATEA MEDIE TRI-
50 REM TRIMESTRIALĂ (M)
55 DIM C(8,12)
60 PRINT "INDICAȚI NUMĂR ARTICOLE (MAXIMUM 8)";
65 INPUT Z
70 FOR K=1 TO Z
72   PRINT "INDICAȚI CANTITĂȚILE LUNARE DE JUCĂRII"
75   FOR I=1 TO 12
80     PRINT "C("; K; " , " ; I; ")=";
85     INPUT C(K, I)
90   NEXT I
95 NEXT K
100 PRINT
105 FOR K=1 TO Z
110 REM CALCUL CANTITATE ANUALĂ JUCĂRII (T)
115 T=0
120 FOR I=1 TO 12
125   T=T+C(K, I)
130 NEXT I
135 REM CALCUL CANTITATE MEDIE TRIMESTRIALĂ (M)
140 M=T/4
145 REM CALCULEAZĂ ȘI AFIȘEAZĂ RIND PENTRU
150 REM TRIMESTRUL 1
155 S=C(K,1)+C(K,2)+C(K,3)
160 R1=S*100/T
165 R2=(S-M)*100/M
```

```
170 PRINT "I", S, R1, R2
175 REM CALCULEAZĂ ŞI AFIŞEAZĂ RIND PENTRU
180 REM TRIMESTRUL 2
185 S=C(K,4)+C(K,5)+C(K,6)
190 R1=S*100/T
195 R2=(S-M)*100/M
200 PRINT "II", S, R1, R2
205 REM CALCULEAZĂ ŞI AFIŞEAZĂ RIND PENTRU
210 REM TRIMESTRUL 3
215 S=C(K,7)+C(K,8)+C(K,9)
220 R1=S*100/T
225 R2=(S-M)*100/M
230 PRINT "III", S, R1, R2
235 REM CALCULEAZĂ ŞI AFIŞEAZĂ RIND PENTRU
240 REM TRIMESTRUL 4
245 S=C(K,10)+C(K,11)+C(K,12)
250 R1=S*100/T
255 R2=(S-M)*100/M
260 PRINT "I V ", S, R1, R2
265 PRINT
270 PRINT
275 NEXT K
280 END
RUN

INDICAŢI NUMĂR ARTICOLE (MAXIMUM 8) ? 2

INDICAŢI CANTITĂŢILE LUNARE DE JUCĂRII

C(1,1)=? 5          C(2,1)=? 50
C(1,2)=?10          C(2,2)=?100
C(1,3)=?15          C(2,3)=?150
C(1,4)=?20          C(2,4)=?200
C(1,5)=?25          C(2,5)=?250
C(1,6)=?30          C(2,6)=?300
C(1,7)=?35          C(2,7)=?350
C(1,8)=?40          C(2,8)=?400
C(1,9)=?45          C(2,9)=?450
C(1,10)=?50         C(2,10)=?500
C(1,11)=?55         C(2,11)=?550
C(1,12)=?60         C(2,12)=?600

I        30     7.6923     -69.2307
II       75     19.2307    -23.0769
III     120     30.7692     23.0769
IV      165     42.3076     69.2307

I       300     7.62923    -69.2307
II      750     19.2307    -23.0769
III    1200     30.7692     23.0769
IV     1650     42.3076     69.2307
```

Remarci

- În linia 55 s-a definit matricea C (cu 8 linii şi 12 coloane)
- Cantităţile lunare de jucării se introduc dinamic în matricea C, cu instrucţiunea **INPUT** din cadrul ciclurilor 70–90şi 75–90 (două bucle **FOR-NEXT**).
- Cantitatea medie trimestrială se calculează cu instrucţiunea

140 M=T/4

în care T (cantitatea anuală de jucării) a fost calculat cu secvenţa de instrucţiuni 105–130.

- Valorile lui R1 şi R2 se calculează cu relaţiile:

R1=S*100/T,
R2=(S–M)*100/M

în care S se determină pentru fiecare trimestru în parte cu instrucţiunile de atribuire corespunzătoare (v. liniile 155, 185, 215, 245).

- Deoarece instrucţiunile 160, 165; 190, 195; 220, 225 şi 250, 255 sînt identice se recomandă gruparea acestora într-o subrutină ce urmează a fi apelată din punctele respective.

☐ **Programul pentru calculul abaterii consumurilor trimestriale** raportate faţă de consumurile trimestriale normate ale mai multor articole este:

```
10 PRINT ' PROGRAMUL CITEŞTE CONSUMURILE TRIMESTRIALE NORMATE (N)'
15 PRINT ' ŞI CONSUMURILE TRIMESTRIALE RAPORTATE (L) ALE MAI '
20 PRINT ' MULTOR ARTICOLE (Z), CALCULEAZĂ ŞI TIPĂREŞTE ABATEREA (A) '
25 PRINT ' FAŢĂ DE CONSUMURILE TRIMESTRIALE NORMATE '
30 PRINT
35 PRINT
40 PRINT
45 DIM C(8), L(4,3), A(8,4), N(4)
50 PRINT ' INDICAŢI NUMĂRUL DE ARTICOLE '
55 INPUT Z
60 PRINT 'INDICAŢI CODURILE ARTICOLELOR'
65 FOR I=1 TO Z
70     INPUT C(I)
75 NEXT I
80 FOR K=1 TO Z
85     PRINT "INDICAŢI CONSUMURILE TRIMESTRIALE NORMATE PENTRU"
90     PRINT "ARTICOLUL CU CODUL:", C(K)
95   FOR I=1 TO 4
100    INPUT N(I)
105  NEXT I
110  PRINT "INDICAŢI CONSUMURILE TRIMESTRIALE RAPORTATE PENTRU"
115  PRINT "ARTICOLUL· CU CODUL" C(K)
120  FOR I=1 TO 4
125    FOR J=1 TO 3
130      INPUT L (I, J)
135    NEXT J
140  NEXT I
145  REM 'CALCUL ABATERE'
150  FOR I=1 TO 4
155     FOR J=1 TO 3
160        IF J= < > 1 HEN 170
165           T=0
170        T=T+L(I, J)
175        IF J < > 3 THEN 185
180        A(K, I)=(T–N(I)) * 100/N(I)
185     NEXT J
190   NEXT I
195   NEXT K
200   REM 'TIPĂRIREA CAP TABEL'
205   PRINT '  ABATERE %  '
210   PRINT   '
215   PRINT ' COD TRIMESTRUL 1 TRIMESTRUL 2 TRIMESTRUL 3 TRIMESTRUL 4'
220   PRINT
225   PRINT
230   PRINT
235   REM ' TIPĂRIRE REZULTATE '
240     FOR I=1 TO Z
```

```
245    PRINT C(I),
250    FOR J=1 TO 4
255       PRINT A (I, J),
260    NEXT J
265    NEXT I
270    STOP
275    END
```

```
RUN
INDICAȚI NUMĂRUL DE ARTICOLE ? 8
INDICAȚI CODURILE ARTICOLELOR ? 1, 2, 3, 4, 5, 6, 7, 8
INDICAȚI CONSUMURILE TRIMESTRIALE NORMATE PENTRU
ARTICOLUL CU CODUL 1
5
10
15
20
INDICAȚI CONSUMURILE TRIMESTRIALE RAPORTATE PENTRU
ARTICOLUL CU CODUL 1
3
8
.
.
.

                        ABATERE %

COD  TRIMESTRUL 1  TRIMESTRUL 2  TRIMESTRUL 3  TRIMESTRUL 4
.         .             .              .            .
.         .             .              .            .
.         .             .              .            .
3         885           60             58.4615      16.1538
4         1912          1155.25        170.769      433.338
.         .             .              .            .
.         .             .              .            .
.         .             .              .            .
```

Remarci

- În linia 45 s-au definit vectorii C, N și matricile L (4 linii și 3 coloane), A (8 linii și 4 coloane).
- De reținut procedura de calcul a abaterii pe care o vom decupa din programul principal.

```
150 FOR I=1 TO 4
155    FOR J=1 TO 3
160       IF J < > 1 THEN 170
165          T=0
170          T=T+L(I, J)
175          IF J < > 3 THEN 185
180          A(K, I)=(T-N (I)) * 100/N (I)
185    NEXT J
190 NEXT I
```

- Pentru înțelegerea programului recomandăm **simularea execuției acestuia cu** creionul și hîrtia. Cel care vă va verifica va fi evident calculatorul cu care lucrați.

CONVERSAȚIA 7

Proiectarea și realizarea unui program BASIC pentru crearea unui fișier secvențial de livrări. Instrucțiuni BASIC-AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS, ABASIC pentru crearea fișierelor de organizare secvențială. Programe utile pentru gestionarea fondurilor de date în fișiere. JOCURI, APLICAȚII și TESTE pentru cititor



□ De la problemă la program

Pînă acum, toate datele folosite de către programele noastre au fost introduse prin instrucţiunile **INPUT, READ-DATA** sau **LET.** Cu consecvenţă, am depozitat datele ca parte integrantă a programului sau le-am introdus direct de la tastatură. Această manieră de lucru se pretează foarte bine pentru un volum mic de date pe care dorim să-l prelucrăm o singură dată. În cazul în care se doreşte prelucrarea unui volum mare de date sau este necesară întreţinerea acestui fond de date se impune ca datele să fie separate de program structurîndu-le în aşa-numitele **fişiere de date.**

Ce este un fişier? **Fişierele** reprezintă un mijloc simplu de a manevra structuri de date, ce pot fi citite şi întreţinute prin programe (BASIC). În general, un fişier poate fi asociat cu o mulţime organizată de informaţii.

Utilizarea fişierelor permite modificarea, sortarea, interclasarea datelor în concordanţă cu specificaţiile de programare. Posibilităţile sînt limitate numai prin capacitatea noastră de a gîndi modul de rezolvare informatică a problemelor.

Fişierele de date sînt folosite pentru gestiunea stocurilor, sistemele de producţie, sistemul de personal, proiectarea asistată de calculator, într-un cuvînt în majoritatea activităţilor profesionale.

Una din trăsăturile acestor fişiere care le fac „misterioase" este aceea că sînt invizibile. Prelucrarea datelor structurate în fişiere nu presupune şi listarea de fiecare dată a conţinutului acestora, deşi este bine ca programele să furnizeze şi anumite liste de control pentru a putea urmări ceea ce prelucrează calculatorul. După ce veţi vedea cîteva exemple, vă veţi putea convinge că într-adevăr calculatorul efectuează operaţiile dorite.

Ceea ce trebuie reţinut este faptul că putem introduce/extrage sau modifica datele organizate în fişiere sub controlul programelor. Fişierele BASIC pot fi **secvenţiale** (un şir continuu de articole, memorate fizic pe suport în ordinea în care au fost scrise de programator – v. BASIC-AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS, ABASIC), **masive virtuale** (masive de date rezidente pe un fişier disc – v. BASIC-PLUS), **relative** (v. BASIC-COMMODORE) şi **selective** (în acces direct, v. BASIC-AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS – fişiere articol).

Pentru a ne familiariza cu modul în care datele pot fi introduse într-un fişier vă propunem să proiectăm şi să realizăm un program scurt, folositor, prin care să creăm pe un suport magnetic (dischetă, disc) fişierul privind livrările de benzină ale unei staţii de benzină şi uleiuri efectuate pentru diverşi beneficiari.

Se vor introduce în fişier datele privind: codul beneficiarului, cantitatea de benzină livrată şi preţul în lei pe tona de benzină. Numărul beneficiarilor nu este dinainte cunoscut. Convenim ca introducerea datelor în fişier să fie oprită în momentul cînd tastăm pentru codul beneficiarului valoarea 99 (un cod artificial).

Limbajul pe care îl vom utiliza este analog celui folosit pentru un fişier manual: **deschide** fişierul, **scrie** un element, **închide** fişierul.

De notat că toate elementele unei înregistrări a fişierului privind: codul, cantitatea livrată şi preţul se scriu succesiv pe dischetă separate prin virgulă.

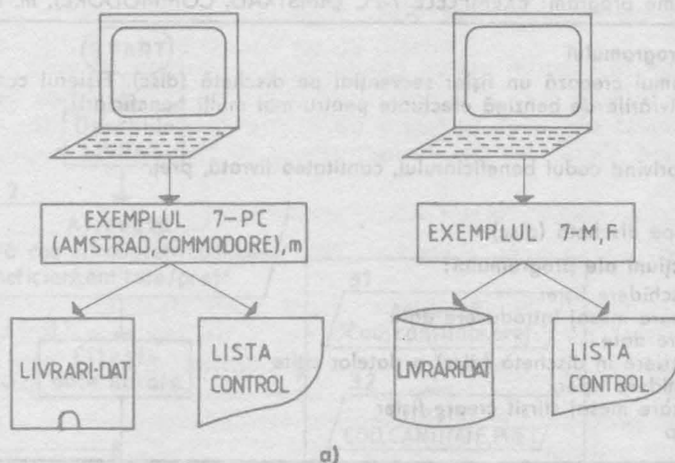
□ Specificaţii de programare

Schema de sistem a EXEMPLELOR 7-PC (AMSTRAD, COMMODORE), m, M, F este ilustrată în modulele de analiză şi proiectare structurată, figura 7.1. Programul citeşte de la tastatură codul beneficiarului, cantitatea livrată, preţul tonei de benzină şi le introduce în fişierul LIVRĂRI. DAT de pe dischetă/disc. De remarcat prezenţa LISTEI DE CONTROL ce conţine înregistrările fişierului creat.

Structura înregistrării de ieşire. O înregistrare a fişierului LIVRĂRI. DAT este structurată logic în trei cîmpuri: cod-beneficiar, cantitate-livrată, preţ-tonă-benzină. Datele sînt separate fizic prin delimitatorul ",".

Datele de intrare se transmit de la tastatură.

Tabela de variabile, specificaţiile de programare şi alocarea funcţiunilor de prelucrare sînt definite în modulele de analiză şi proiectare structurată, figura 7.1.



a)

FORMATUL DATELOR DE IEŞIRE

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

COD	CANT	PREŢ

b)

Fig. 7.1. Modulele de analiză şi proiectare structurată: *a)* schema de sistem; *b)* formatul datelor de ieşire;

DATE INTRARE		
Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE) m, M, F		
COD BENEFICIAR	CANTITATE BENZINĂ LIVRATĂ (tone)	PREŢ TONĂ (lei)
10	20	8.000
30	30	7.500
20	40	8.000
99	99	99

c)

TABELA DE VARIABILE		
Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F		
Variabile de intrare	Variabile de stare	Variabile de ieşire
COD: cod beneficiar CANT: cantitate livrată PREŢ: preţ tonă benzină		COD: cod beneficiar CANT: cantitate livrată PREŢ: preţ, tonă benzină

d)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

Descrierea programului

Programul creează un fişier secvenţial pe dischetă (disc). Fişierul conţine date referitoare la livrările de benzină efectuate pentru mai mulţi beneficiari.

Intrări

Date privind codul beneficiarului, cantitatea livrată, preţ.

Ieşiri

Fişier pe dischetă (disc).

Lista de funcţiuni ale programului:

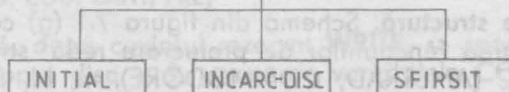
1. Deschidere fişier
2. Afişare mesaj introducere date
3. Citire date
4. Inscriere in dischetă (disc) a datelor citite
5. Inchidere fişier
6. Afişare mesaj sfirşit creare fişier
7. Stop

e)

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE	
Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F	
Modul	Funcţiuni
INIŢIAL	1, 2, 3
INCARC-DISC	4, 3
SFIRŞIT	5, 6, 7

f)

Fig. 7.1. *c)* date intrare; *d)* tabela de variabile; *e)* specificaţii de programare; *f)* alocarea funcţiunilor de prelucrare;

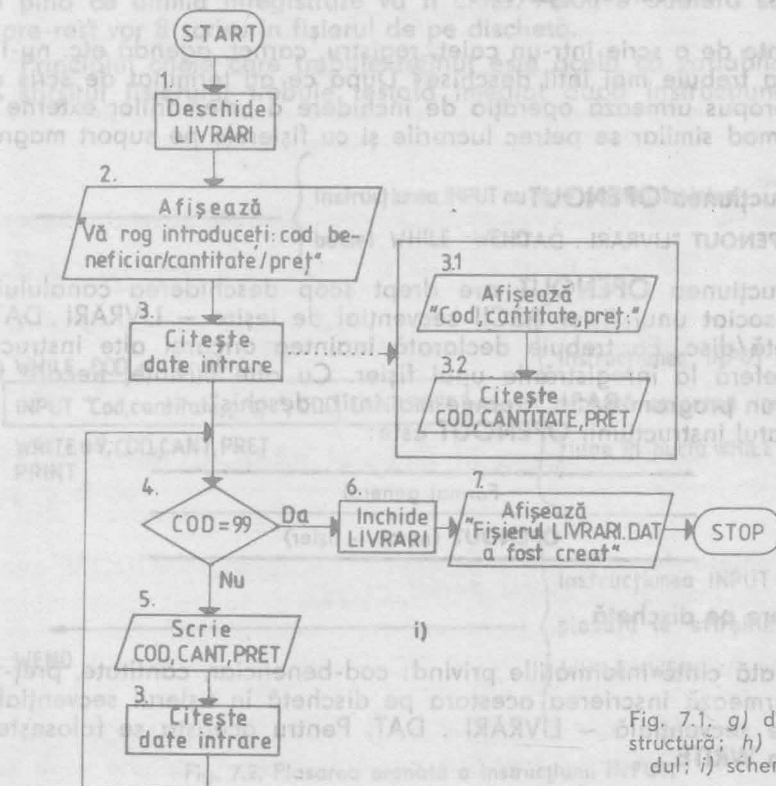


g)

PSEUDOCODUL	
Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F	

```
EXEMPLUL7      SEQ
INIȚIAL        SEQ
               Deschide LIVRĂRI
               Afișează "Vă rog introduceți: cod beneficiar/cantitate/preț"
               Afișează un rind de spații
               Citește date intrare
INIȚIAL        END
INCARC-DISC    CIT TIMP COD <> 99
                   Scrie COD, CANT, PREȚ
                   Citește date intrare
INCARC-DISC    SFÎRȘIT
               Inchide LIVRĂRI
               Afișează "Fișierul LIVRĂRI. DAT a fost creat"
EXEMPLUL7      END
```

h)



i)

Fig. 7.1. *g)* diagrama de structură; *h)* pseudocodul; *i)* schema logică.

☐ Documentaţia de proiectare

Diagrama de structură. Schema din figura 7.1 (g) completată cu tabelul pentru alocarea funcţiunilor de prelucrare redă structura logică a EXEMPLELOR 7-PC (AMSTRAD, COMMODORE), m, M, F. De notat prezenţa unor funcţiuni noi: deschidere fişiere, închidere fişiere.

Pseudocodul şi schema logică. Pseudocodul pentru EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F şi schema logică sînt prezentate în modulele de analiză şi proiectare structurată, fig. 7.1.

Observaţii

- Orice fişier BASIC, înainte de a fi prelucrat, trebuie deschis iar după aceea închis.
- Pentru crearea unui fişier BASIC se recomandă utilizarea structurii de iteraţie **CÎT TIMP** (v. cazul fişierelor vide).

☐ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 227

Programul BASIC este prezentat în volumul 2, pagina 227. În cele ce urmează vom analiza numai instrucţiunile din liniile 30, 80 şi 120.

OPEN – Deschideţi fişierele!

Înainte de a scrie într-un caiet, registru, carnet, agendă etc. nu-i aşa că acestea trebuie mai întîi deschise? După ce aţi terminat de scris ceea ce v-aţi propus urmează operaţia de închidere a „memoriilor externe" folosite. În mod similar se petrec lucrurile şi cu fişierele pe suport magnetic.

Instrucţiunea OPENOUT

```
30 OPENOUT "LIVRĂRI . DAT"
```

Instrucţiunea **OPENOUT** are drept scop deschiderea canalului de transfer asociat unui fişier ASCII secvenţial de ieşire – LIVRĂRI . DAT de pe dischetă/disc. Ea trebuie declarată înaintea oricărei alte instrucţiuni care se referă la înregistrările unui fişier. Cu alte cuvinte, fiecare fişier folosit de un program BASIC trebuie mai întîi „deschis".

Formatul instrucţiunii **OPENOUT** este:

Format general
OPENOUT ⟨nume – fişier⟩

Scriere pe dischetă

O dată citite informaţiile privind: cod-beneficiar, cantitate, preţ-tonă benzină urmează înscrierea acestora pe dischetă în fişierul secvenţial, cu organizare secvenţială – LIVRĂRI . DAT. Pentru aceasta se foloseşte instrucţiunea **WRITE.**

Instrucţiunea WRITE

80 **WRITE** # 9, COD, CANT, PREŢ

De notat că, după cuvîntul rezervat **WRITE,** se specifică numărul canalului pentru floppi disc – 9, urmat de variabilele COD, CANT şi PREŢ separate prin virgule.

Formatul instrucţiunii **WRITE** este:

Format general
WRITE [# ⟨nr. canal⟩,][⟨listă expresii⟩]

Tehnica plasării instrucţiunii INPUT

Cînd întîlniţi un program structurat este bine să studiaţi cu atenţie plasarea şi manevrarea instrucţiunii **INPUT.** Întrebările obişnuite privind plasarea instrucţiunii **INPUT** pentru programul BASIC-AMSTRAD pot fi: „De ce s-a plasat o instrucţiune **INPUT** o dată înainte de bucla **WHILE-WEND** şi apoi la sfîrşitul acesteia?" „De ce nu plasăm instrucţiunea **INPUT** doar la începutul buclei **WHILE-WEND?**

În figura 7.2 este prezentată imaginea unui astfel de program.

Iată răspunsul: un program scris în această manieră va lucra foarte bine pînă ce ultima înregistrare va fi citită. Apoi, o dublură sau o înregistrare-rest vor fi scrise în fişierul de pe dischetă.

Principiul cheie care trebuie reţinut este acela că variabila care indică sfîrşitul fişierului trebuie testată imediat după instrucţiunea **INPUT**

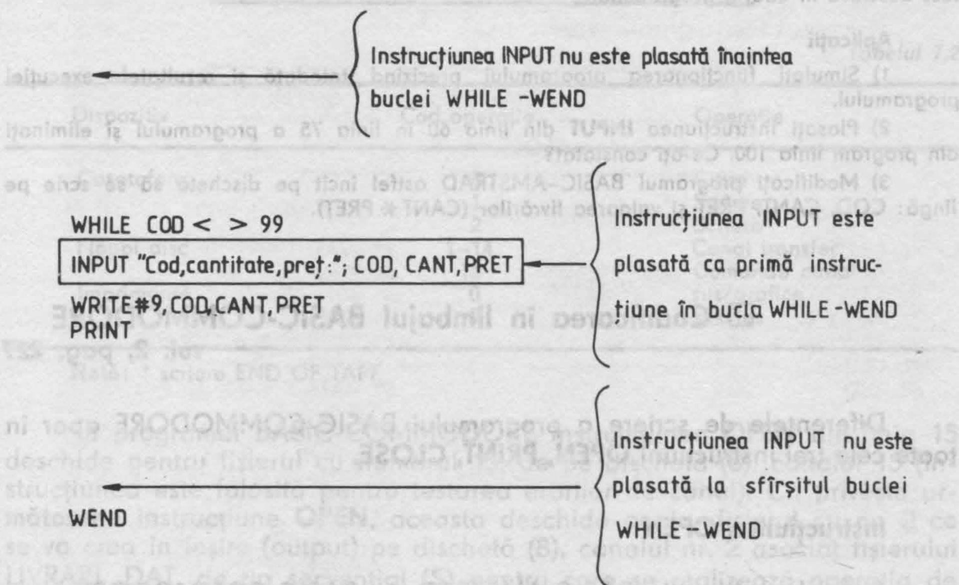


Fig. 7.2. Plasarea eronată a instrucţiunii **INPUT.**

(operaţie de citire). În cadrul programului, variabila COD este testată de instrucţiunea **WHILE** COD<>99 (linia 70).

Prin urmare, înaintea primei prelucrări, o înregistrare trebuie deja să fie citită. Acest **INPUT** iniţial a fost plasat apoi la sfîrşitul buclei **WHILE-WEND.**

Observaţie. Acesta nu este singurul mod de plasare a instrucţiunii **INPUT,** dar el este relativ uşor de înţeles şi utilizat pentru programele BASIC simple. Într-o asemenea manieră vom întrebuinţa şi noi tehnica plasării instrucţiunii **INPUT.**

CLOSE – închideţi fişierele!

În momentul în care s-a citit sfîrşitul fişierului de date (COD=99), controlul programului este trecut la instrucţiunea:

```
120 CLOSEOUT
```

care realizează închiderea fişierului de ieşire LIVRĂRI . DAT.

Instrucţiunea CLOSEOUT

Instrucţiunea închide canalul de transfer asociat fişierului de ieşire (LIVRĂRI . DAT). Formatul instrucţiunii este:

Format general
CLOSEOUT

Observaţie. Instrucţiunea **CLOSEOUT** se execută numai pentru fişierele care au fost deschise în cadrul programului.

Aplicaţii

1) Simulaţi funcţionarea programului precizînd totodată şi rezultatele execuţiei programului.

2) Plasaţi instrucţiunea **INPUT** din linia 60 în linia 75 a programului şi eliminaţi din program linia 100. Ce-aţi constatat?

3) Modificaţi programul BASIC–AMSTRAD astfel încît pe dischetă să se scrie pe lîngă: COD, CANT, PREŢ şi valoarea livrărilor (CANT ✱ PREŢ).

□ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 227

Diferenţele de scriere a programului BASIC-COMMODORE apar în toate cele trei instrucţiuni **OPEN, PRINT, CLOSE.**

Instrucţiunea OPEN

Spre deosebire de BASIC-AMSTRAD, în BASIC-COMMODORE pentru instrucţiunea **OPEN** se specifică numărul fişierului (poate avea valori

între 1 şi 255), numărul dispozitivului (8 pentru floppi disc), numărul canalului de transfer (de la 2 la 14 inclusiv) şi numele fişierului.

```
15 OPEN 15, 8, 15
20 OPEN 2, 8, 2, "LIVRĂRI.DAT, S, W"
```

Formatul instrucţiunii este:

Format general
OPEN ⟨nr. fişier⟩, ⟨dispozitiv⟩, ⟨nr. canal⟩, "⟨nume-fişier, tip, R/W⟩"

De notat că la deschiderea unui fişier secvenţial, cel de-al patrulea parametru are în structura sa nume fişier, tip fişier: **P** pentru program (PRG); **S** pentru fişier secvenţial (SEQ); **R** pentru fişier relativ (REL), operaţie: **R** (READ) sau **W** (WRITE).

În tabelul 7.1 sînt prezentate dispozitivele periferice şi codurile asociate acestora iar în tabelul 7.2 sînt ilustrate operaţiile I/0 şi codurile corespunzătoare pentru fiecare dispozitiv în parte.

Tabelul 7.1

Dispozitiv	Cod dispozitiv
Tastatură	0
Casetofon	1
RS-232	2
Monitor	3
Imprimantă	4
Imprimantă	5
Floppi disc	8

Tabelul 7.2

Dispozitiv	Cod operaţie	Operaţie
Casetofon	0	Citire
	1	Scriere
	2	Scriere*
Floppi disc	1–14	Canal transfer
	15	Comandă canal
Imprimantă	0	sus/grafice
	7	sus/jos

Notă: * scriere END OF TAPE

În programul BASIC-COMMODORE instrucţiunea **OPEN** din linia 15 deschide pentru fişierul cu numărul 15, de pe dischetă (8), canalul 15 (instrucţiunea este folosită pentru testarea erorilor de canal). Cît priveşte următoarea instrucţiune **OPEN**, aceasta deschide pentru fişierul cu nr. **2** ce se va crea în ieşire (output) pe dischetă (8), canalul nr. 2 asociat fişierului LIVRĂRI . DAT, de tip secvenţial (**S**) pentru care se realizează operaţia de scriere **W (WRITE)**.

Instrucţiunea PRINT #

În BASIC-COMMODORE, scrierea datelor într-un fişier secvenţial se realizează cu instrucţiunea **PRINT #**.

100 **PRINT** # 2, K%; A$; C2; A$; C3

PRINT # nu compactează datele, ci le scrie aşa cum ar fi afişate de **PRINT**. Pentru a obţine un cîmp compact de date în fişier se utilizează ca delimitator în ⟨lista⟩ caracterul " ; ". Dacă se utilizează " , " se scriu pe disc şi spaţiile care ar apare la afişare.

Pentru a forma explicit şirurile pe disc, acestea trebuie explicit separate prin delimitatori (" ; "; " , ", "[CR]").

Formatul instrucţiunii este:

Format general
PRINT # ⟨nr. fişier⟩ [,⟨listă⟩]

Instrucţiunea CLOSE

În BASIC-COMMODORE închiderea unui fişier se realizează cu instrucţiunea **CLOSE** urmată de numărul fişierului declarat în instrucţiunea **OPEN**.

120 **CLOSE** 2 : **CLOSE** 15

Formatul instrucţiunii este:

Format general
CLOSE ⟨nr. fişier⟩

Aplicaţie. Introduceţi şi executaţi următoarele programe:

1.
```
 5 OPEN 15, 8, 15
10 OPEN 5, 8, 5, "STUDENT, S, W"
20 A$="DANIEL" : B$="ANDREI"
30 PRINT # 5, A$
40 PRINT # 5, B$
50 CLOSE 5, 15
60 END
```

2.
```
 5 OPEN 15, 8, 15
10 OPEN 5, 8, 5, "STUDENT", S, W"
20 A$="DANIEL" : B$="ANDREI"
30 X$=CHR$ (13)
40 PRINT # 5, A$, X$, B$
50 CLOSE 5, 15
```

3.
```
10 FOR I=1 TO 6
20 READ B$ (I)
30 READ CANT (I)
40 READ PREŢ (I)
50 V(I)=CANT(I) * PREŢ(I)
60 NEXT I
70 DATA P-1, 30, 20, P-2, 50, 60,
   P-3, 20, 10
80 DATA P-4, 15, 20, P-5, 16, 10,
   R-6, 3, 9
85 X$=CHR$ (13)
90 OPEN 15, 8, 15
100 OPEN 2, 8, 2, "AX1, S, W"
110 FOR I=1 TO 6
120 PRINT # 2, B$(I); X$; CANT(I);
    X$; PREŢ(I); X$; V(I)
130 NEXT I
140 CLOSE 2 : CLOSE 15
150 END
```

□ Particularităţi ale programării în limbajul BASIC-80

vol. 2, pag. 228

Faţă de programele din exemplele precedente apar diferenţe la scrierea instrucţiunilor **OPEN, PRINT** şi **CLOSE.**

Instrucţiunea OPEN

30 **OPEN** "O", # 1, "LIVRĂRI.DAT"

Instrucţiunea **OPEN** alocă un buffer pentru operaţiile de I/0 şi determină modul de acces. Formatul instrucţiunii este:

Format general
OPEN "⟨MOD⟩", [#]⟨nr. fişier⟩, "⟨nume-fişier⟩", [⟨lung-înreg⟩]

unde, ⟨MOD⟩ este un şir de expresii în care primul caracter poate fi: O (mod de ieşire secvenţial); I (mod de intrare secvenţial); R (mod I/0 în acces direct);

⟨nr. fişier⟩ este o expresie întreagă cu valori de la 1 la 15. Numărul este asociat fişierului pe toată perioada cît acesta este deschis şi folosit în instrucţiunile de I/0;

⟨lung-înreg⟩ este o expresie care, atunci cînd este precizată, stabileşte lungimea înregistrării pentru operaţii de I/0 în acces direct (implicit, lungimea unei înregistrări se consideră 128 octeţi).

Observaţie. Într-un program BASIC-80 acelaşi fişier poate fi deschis sub mai multe numere pentru citire secvenţială sau acces direct, dar numai cu un singur număr pentru scriere secvenţială.

Instrucţiunea PRINT

80 **PRINT** #1, COD; " , " ; CANT; " , " ; PREŢ

PRINT # scrie date într-un fişier secvenţial BASIC-80. Formatul instrucţiunii este:

Format general
PRINT #⟨nr. fişier⟩[**USING**⟨expresie-şir⟩ ;]⟨listă-expresii⟩

în care:

⟨nr. fişier⟩ este numărul folosit la deschiderea fişierului;

⟨expresie-şir⟩ (v. instrucţiunea **PRINT USING**);

Remarci

- **PRINT #** nu compactează datele;
- Pentru a forma explicit şirurile pe disc, acestea trebuie explicit separate prin delimitatori.

TEST

Precizaţi rezultatul execuţiei următoarelor programe:

a)
```
10 OPEN "O", # 1, " xx "
20 READ A$, B$
30 DATA CELEBRUL, 702
40 PRINT # 1, A$; B$
50 END
```

b)
```
10 OPEN "O", # 1, " yy "
20 DATA CELEBRUL, 702
30 READ A$, B$
40 PRINT #1, A$; " , "; B$
50 END
```

R.

a) CELEBRUL702
b) CELEBRUL,702

Instrucţiunea CLOSE

```
120 CLOSE # 1
```

Instrucţiunea închide fişierul #1 de pe dischetă.
Formatul instrucţiunii este:

Format general
CLOSE [#]⟨nr. fişier⟩[,[#]⟨nr. fişier⟩ ...⟩]]

Remarci

- ⟨nr. fişier⟩ este numărul sub care fişierul a fost deschis.
- **CLOSE** fără argumente închide toate fişierele deschise.
- Fişierul poate fi deschis sub acelaşi număr sau sub alt număr.
- Instrucţiunile **END** şi **NEW** închid automat toate fişierele (instrucţiunea **STOP** face excepţie).

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 OPEN "O", # 1, "AGENDA"
20 INPUT "NUME"; N$
30 IF N$=" " THEN CLOSE : END
40 INPUT "TELEFON" : " ; T$
50 PRINT #1, N$; " , " ; T$
60 GO TO 20
```

Observaţie. După cum remarcaţi, programul nu este scris structurat. Varianta structurală a acestuia este:

```
10 OPEN "O", # 1, "AGENDA"
20 INPUT "NUME : " ; N$
30 WHILE N$ < > " "
40     INPUT "TELEFON : " ; T$
50     PRINT #1, N$; " , " ; T$
60     INPUT "NUME : " ; N$
70 WEND
80 END
```

□ Particularităţi ale programării în limbajul BASIC-PLUS

vol. 2, pag. 228

Fişiere BASIC—PLUS

Din punct de vedere al structurii şi metodei de acces fişierele BASIC—PLUS pot fi: fişiere secvenţiale ASCII, fişiere masive virtuale şi fişiere articol.

Pentru a înţelege mai bine clasificarea de mai sus va trebui să definim noţiunea de *articol*. Prin *articol* înţelegem elementul unui fişier ce se

poate trata şi identifica (logic/fizic) ca o entitate. De notat că suportul fizic extern al fişierelor BASIC–PLUS poate fi: banda perforată, cartela perforată, hîrtia de imprimantă, ecranul monitoarelor, banda magnetică, discul magnetic. Suporturile de informaţie pot fi orientate fie pe articol (ex. linia de imprimantă), fie pe fişier (disc, bandă magnetică).

Fişierele secvenţiale ASCII reprezintă cea mai simplă metodă de structurare a datelor pe un suport extern. Datele din structura acestor fişiere sînt reprezentate în format ASCII fiind separate între ele (în interiorul unui articol) prin virgulă. De notat că un articol dintr-un fişier ASCII secvenţial este identic cu articolul creat de utilizator în mod conversaţional, printr-o instrucţiune **INPUT.**

Pentru exploatarea fişierelor ASCII secvenţiale puteţi folosi următorul set de instrucţiuni BASIC-PLUS: **OPEN, CLOSE, INPUT#, INPUT LINIE#** (instrucţiune de citire a unui articol ca o singură dată), **PRINT#, PRINT#. . . USING, MAT INPUT#** (instrucţiune matricială de citire a unui masiv dintr-un fişier), **MAT PRINT** (instrucţiune matricială de scriere a unui masiv pe un fişier), **KILL** (instrucţiune de eliminare a unui fişier ASCII secvenţial).

Fişierele masive virtuale sînt masive rezidente pe un fişier disc, ca extensie a memoriei interne, ale căror dimensiuni [26] sînt limitate numai de mărimea spaţiului disponibil pe discul respectiv.

Pentru exploatarea fişierelor masive virtuale puteţi folosi următorul set de instrucţiuni BASIC-PLUS: **DIM#** (instrucţiune pentru declararea masivelor virtuale), **OPEN . . . AS VIRTUAL** (instrucţiune pentru deschiderea unui fişier virtual), **CLOSE.**

Fişierele articol prezintă particularitatea că un articol poate conţine date de tipuri diferite, indiferent de ordinea lor de dispunere. De notat că în cazul fişierelor articol toate operaţiile I/0 se realizează pe blocuri fizice (512 octeţi).

Pentru exploatarea fişierelor articol puteţi folosi următorul set de instrucţiuni BASIC-PLUS: **OPEN, GET#** (instrucţiune pentru citirea unui articol al fişierului), **PUT#** (instrucţiune pentru scrierea unui articol), **FIELD#** (instrucţiune pentru identificarea dinamică a cîmpurilor din articol), **LSET, RSET** (instrucţiuni de transfer al datei în zona de alocare a variabilei).

Identificarea fişierelor BASIC-PLUS

Un fişier BASIC-PLUS se poate identifica prin: **nume extern** (specificator de fişier) şi **nume intern** (număr de fişier). Specificatorul de fişier asigură identificarea fişierului pe suportul extern de către sistemul de operare, iar numărul de fişier asigură identificarea fişierului de către programul BASIC.

Formatul general al unui specificator de fişier [26] este:

Format general
⟨Suport⟩ : [⟨grup⟩, ⟨membru⟩]⟨nume⟩ · ⟨tip⟩ ; ⟨versiune⟩

unde:

⟨suport⟩ reprezintă unitatea periferică fizică (logică) pe care s-a încărcat volumul ce conţine fişierul;

⟨grup⟩, ⟨membru⟩ reprezintă codul utilizator care precizează catalogul în care este introdus fişierul pe suport;

⟨nume⟩ este numele fişierului sub care acesta este identificat pe suport;

⟨tip⟩ precizează natura conţinutului fişierului;

⟨versiune⟩ indică numărul de versiune al fişierului care se exploatează.

Reguli

- Numele pentru ⟨suport⟩ este alcătuit din două caractere alfabetice urmate, opţional, de un număr octal de una sau două cifre: CR (cititor de cartele), DK (disc), MM (banda magnetică), LP (imprimantă), TT (terminal).
- Codurile pentru ⟨grup⟩, ⟨membru⟩ sînt numere octale în intervalul 0–377.
- ⟨nume⟩ este alcătuit din 9 caractere alfanumerice.
- ⟨tip⟩ este format din 3 caractere: **BAC** (fişier program în format sursă), **DAT** (fişier de date), **TMP** (fişier temporar ce se elimină la încheierea sesiunii utilizatorului).
- ⟨versiune⟩ este un număr octal cu valoarea cuprinsă între 0 şi 7777. În cazul în care nu se precizează se consideră: numărul de versiune maxim existent (pentru fişierele de intrare), numărul de versiune maxim existent plus o unitate (pentru fişierele de ieşire).

Observaţie. Pentru suporturile orientate pe articol, specificatorul de fişier trebuie să conţină numai parametrul ⟨suport⟩ (numele suportului) restul informaţiilor fiind ignorate.

Înapoi la program

Programul este prezentat în volumul 2, pagina 228. În cele ce urmează vom face referiri numai asupra instrucţiunilor **OPEN, PRINT#** şi **CLOSE.**

Instrucţiunea OPEN

```
20 OPEN "LIVRĂRI.DAT" FOR OUTPUT AS ASCII & FILE 1 TO WRITE
```

Instrucţiunea **OPEN** serveşte la deschiderea unui canal de transfer asociat fişierului (v. specificatorul de fişier). De asemenea, prin intermediul instrucţiunii **OPEN** se realizează corespondenţa dintre numele extern şi numele intern ale fişierului.

Formatul instrucţiunii este:

Format general
OPEN ⟨specificator-fişier⟩ **[FOR INPUT | OUTPUT] AS [ASCII] FILE** ⟨nr. canal⟩ **TO [READ | WRITE | APPEND]**

unde:

⟨specificator-fişier⟩ este orice expresie de tip şir, avînd ca valoare un nume extern de fişier. Se plasează între apostrofuri;

⟨nr. canal⟩ reprezintă numărul canalului de transfer ce se asociază fişierului. Poate fi orice expresie numerică întreagă cuprinsă între 1 şi 8 inclusiv.

Reguli

- **READ** (v. formatul general) indică deschiderea fişierului pentru citire. Ulterior se poate folosi numai instrucţiunea **INPUT** (fişiere ASCII).
- **WRITE** indică deschiderea fişierului pentru scriere. Ulterior se poate folosi numai instrucţiunea **PRINT** (fişiere ASCII).

- **APPEND** indică deschiderea unui fişier numai la scriere prin extinderea fişierului (fişiere ASCII).
- **FOR INPUT** presupune deschiderea unui canal pentru un fişier deja creat.
- **FOR OUTPUT** presupune deschiderea unui canal pentru crearea unui fişier.
- Instrucţiunea **OPEN** fără **FOR INPUT** sau **FOR OUTPUT** realizează deschiderea unui canal pentru un fişier existent sau pentru crearea unui nou fişier (dacă fişierul specificat nu există).
- Deschiderea canalului pentru prelucrarea unui fişier are ca efect activarea unui buffer de I/O.
- Fişierele ASCII secvenţiale se pot deschide numai în mod **READ, WRITE, APPEND.** Revenind la instrucţiunea **OPEN** din linia 20, remarcaţi că ea serveşte la deschiderea canalului de transfer 1 asociat fişierului ASCII secvenţial. "LIVRĂRI.DAT" deschis în mod **WRITE,** pentru crearea **(FOR OUTPUT)** acestuia.

Instrucţiunea PRINT

```
100 PRINT #1, K%; ","; C2; ","; C3
```

PRINT # realizează transferul datelor din memoria internă a calculatorului pe un suport orientat pe articol sau fişier. De notat că transferul datelor respectă regulile de editare ale instrucţiunii **PRINT.** Formatul instrucţiunii este:

Format general
PRI [NT] # ⟨expresie-numerică⟩ [, ⟨listă⟩]

unde:

⟨expresie-numerică⟩ indică numărul canalului de transfer (0÷8) asociat fişierului;

⟨lista⟩ reprezintă o listă de elemente (admise de limbaj) separate între ele prin " , " sau prin " ; ".

Instrucţiunea CLOSE

```
120 CLOSE 1
```

Execuţia unei instrucţiuni **CLOSE** are ca efect închiderea canalului de transfer deschis prin instrucţiunea **OPEN.**

Formatul instrucţiunii este:

Format general
CLO [SE] ⟨exp. 1⟩ [, ⟨exp. 2⟩] . . .]

unde:

⟨exp. 1⟩, ⟨exp. 2⟩ . . . reprezintă orice expresie numerică a cărei parte întreagă reprezintă un număr de canal cuprins între 1 şi 8 inclusiv.

Reguli

- Orice canal deschis pentru un fişier trebuie în mod obligatoriu închis.
- Absenţa parametrilor ⟨exp. 1⟩, ⟨exp. 2⟩ . . . dintr-o instrucţiune **CLOSE** are ca efect închiderea tuturor canalelor de transfer la acel moment.

Aplicaţie. Să se creeze un fişier ASCII secvenţial care să conţină temperaturile zilnice din mai multe staţii meteorologice. Structura înregistrării fişierului este: Cod – staţie, denumire – staţie, temperatură – luni, temperatură – marţi, ..., temperatură – duminică.

Joc pe HC-85, TIM S, SPECTRUM

```
 10 REM DE LA 1 LA 5
100 DIM a(3, 3)
105 LET c$="123456789"
108 PRINT AT 8, 4; "SELECTAŢI";
    TAB 15; "TABLOU"
110 PRINT AT 10, 6; "—"; TAB 16; "—"
120 FOR i=1 TO 3
125 LET i2=i * 2: LET i3=(i-1) * 3
130 PRINT AT i2+9,5; "I"; i3+1; i3+2;
    "  "; i3+3; "I"; TAB 15; "I I"
135 PRINT AT i2+10,5; "I I"; TAB 15; "I I"
140 NEXT i
150 PRINT AT 16,5; "——"; TAB 15; "——"
160 FOR a=1 TO 10
170 LET i=INT (RND * 9)+1:
    GO SUB 400
180 NEXT a
190 LET m=0: LET t=0
200 LET t$=INKEY$
210 IF t$="" THEN GO TO 200
220 FOR i=1 TO 9
230 IF t$=c$ (i) THEN GO TO 260
240 NEXT i
250 GO TO 200
260 LET m=m+1
270 GO SUB 400
280 IF t<9 THEN GO TO 200
290 CLS: PRINT FLASH 1; "Aţi reuşit!"
300 PRINT : PRINT "IN"; m;
    "ÎNCERCĂRI"
310 STOP
400 LET x=INT ((i-1)/3)+1
410 LET y=i-3 * (x-1)
420 LET a(x, y)=a(x, y)-1
430 FOR i=1 TO 3
440 LET a(x, i)=a(x, i)+1
450 LET a(i, y)=a(i, y)+1
460 NEXT i
470 LET t=0
480 FOR i=1 TO 3
490 FOR j=1 TO 3
500 IF a(i, j)=6 THEN LET a(i, j)=0
510 IF a(i, j)=0 THEN LET t=t+1
520 PRINT AT 11+(i-1) * 2, 16+(j-1)*2;
    a(i, j)
530 NEXT j
540 NEXT i
550 RETURN
```

□ Particularităţi ale programării în limbajul ABASIC

vol. 2, pag. 229

Fişiere ABASIC

Fişierele ABASIC sînt de tip *secvenţial* cu lungimea maximă a înregistrării de 140 octeţi. În număr de cel mult 8, fişierele ABASIC pot fi folosite doar în mod scriere **(PRINT/WRITE)** sau citire **(INPUT/LINPUT)**. Fişierele se identifică printr-un nume precizat în instrucţiunea **OPEN** (respectă convenţiile de sintaxă ARIEL) căruia i se asociază un număr intern (precedat opţional de "#").

De notat că fişierele ABASIC sînt constituite din şiruri de caractere de lungime variabilă, avînd structura liniilor de terminal utilizate.

Înapoi la program

Programul este prezentat în volumul 2, pag. 229. Vom analiza şi de această dată instrucţiunile **OPEN, WRITE#, CLOSE.**

Instrucţiunea OPEN

```
30 OPEN #1, "LIVRĂRI . DAT"
```

Formatul instrucţiunii este:

Format general
OPEN [#]⟨expresie-numerică⟩ "⟨nume-fişier⟩"

unde:

⟨expresie-numerică⟩ reprezintă numărul întreg (1–8) prin care va fi referit fişierul;

⟨nume-fişier⟩ este identificatorul fişierului (şir de caractere plasat între apostrofuri ce respectă convenţiile de sintaxă ARIEL).

Instrucţiunea WRITE

80 **WRITE** #1, COD, CANT, PREŢ

Instrucţiunea are acelaşi efect ca şi cele de scriere directă la terminal (**PRINT, PRINT USING**).

Formatul instrucţiunii este:

Format general
WRITE # ⟨expresie-numerică⟩ [, ⟨listă⟩]

unde: ⟨listă⟩ reprezintă o listă de elemente admise de limbajul ABASIC, separate între ele prin " , " sau prin " ; ".

Instrucţiunea CLOSE

120 **CLOSE** #1

Instrucţiunea închide fişierul #1. Ea are formatul:

Format general
CLOSE [[#]⟨expresie 1⟩ [, ⟨expresie 2⟩ . . .]]

unde prin ⟨expresie 1⟩, ⟨expresie 2⟩ . . . se precizează identificatorul intern al fişierelor a căror închidere este cerută.

Reguli

- **CLOSE** fără parametri are ca efect închiderea tuturor fişierelor deschise prin instrucţiunea **OPEN.**
- Dacă nu se foloseşte **CLOSE,** fişierele deschise în scriere **(PRINT/WRITE)** sînt şterse la execuţia uneia din comenzile: **RUN, LOAD, NEW, EXIT.**

Aplicaţie. Să se creeze un fişier ABASIC necesar generării următorului raport:

AUTOR	TITLUL LUCRĂRII	EDITURA	ANUL APARIŢIEI	COTA CĂRŢII
Patrubani, N.	Microprocesorul Z 80	TEHNICĂ	1989	1-BS-203
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

TEMA 7

☐ Răspundeţi prin DA sau NU la următoarele întrebări:

- Pentru prelucrarea unui volum mic de date de către un program BASIC se utilizează instrucţiunile: **INPUT, LET, READ-DATA.**
- Pentru prelucrarea unui volum mare de date se impune ca datele să fie separate de program şi să se organizeze în fişiere.
- Într-un fişier secvenţial articolele sînt dispuse secvenţial şi devin accesibile tot secvenţial.
- Într-un fişier ASCII secvenţial, datele sînt separate în interiorul unui articol prin punct şi virgulă.
- Într-un fişier se pot introduce date numerice, şiruri de caractere sau combinaţii ale acestora, separate prin spaţii sau virgulă.
- Canalul este unitatea de schimb de informaţie între un fişier şi un program BASIC.
- În BASIC-AMSTRAD numărul canalului pentru unitatea de floppi-disc este 9.
- În BASIC-COMMODORE numărul canalului de transfer are valori de la 2 la 14 inclusiv.
- În BASIC-PLUS numărul canalului de transfer are valori de la 1 la 8 inclusiv.
- Înainte de a scrie într-un fişier, acesta trebuie deschis cu instrucţiunea **OPEN.**
- În limbajul BASIC-AMSTRAD, instrucţiunea **OPENOUT** are drept scop deschiderea canalului de transfer asociat unui fişier de intrare.
- Într-o instrucţiune **OPEN** din limbajul BASIC–COMMODORE se specifică numărul fişierului, numărul dispozitivului, numărul canalului de transfer şi numele fişierului.
- În BASIC-COMMODORE numele fişierului din instrucţiunea **OPEN** nu trebuie să conţină obligatoriu tipul fişierului **(S, R)** şi nici tipul operaţiei **(R, W).**
- În instrucţiunea **OPEN** din BASIC-80 se indică opţional modul I/O prin una din literele: O, I sau R.
- Fişierele ASCII secvenţiale BASIC-PLUS se pot deschide numai în mod **READ, WRITE, APPEND.**
- Fişierele ABASIC sînt de tip secvenţial şi selectiv.
- Instrucţiunea **CLOSE** realizează închiderea unui singur fişier.
- În BASIC-AMSTRAD scrierea datelor într-un fişier secvenţial pe dischetă se realizează cu instrucţiunea **WRITE # 8.**
- În ABASIC scrierea datelor într-un fişier secvenţial pe disc se realizează cu instrucţiunile **PRINT #/WRITE #.**

☐ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Fişierul reprezintă .... Fişierele BASIC pot fi ....
b) Pentru deschiderea unui fişier de ieşire se utilizează instrucţiunea ....

.... în limbajul BASIC-AMSTRAD şi instrucţiunile .... , ... în BASIC-COMMODORE, BASIC-PLUS, BASIC-80, ABASIC.

c) În limbajul BASIC-COMMODORE instrucţiunea

```
10 OPEN 3, 8, 3, "AGA.S, W"
```

indică ....

d) În limbajul BASIC-80 instrucţiunile:

```
10 OPEN "O", #2, "AGA.DAT"
20 OPEN "I", #2, "AGA.DAT"
30 OPEN "R" #2, "AGA.DAT"
```

precizează ....

e) În limbajul BASIC–PLUS instrucţiunile:

```
10 OPEN "A.DAT" FOR OUTPUT AS ASCII & FILE 1 TO WRITE
20 OPEN "A.DAT" FOR INPUT AS ASCII & FILE 1 TO READ
```

precizează ....

f) În limbajul BASIC–PLUS opţiunea **READ** într-o instrucţiune **OPEN** indică .... , iar opţiunea **WRITE** indică ....

g) În BASIC-80, BASIC .... , ... , ... instrucţiunea **OPEN** alocă un buffer pentru operaţiile de I/O.

h) În BASIC-PLUS "numele extern" pentru un fişier asigură identificarea de către sistemul de operare a fişierului pe suportul extern, iar "numele intern" ....

i) În limbajul BASIC–PLUS transferul de date între memoria internă şi un suport orientat pe articol sau fişier se realizează prin instrucţiunea ....

j) În limbajul BASIC–PLUS instrucţiunea **OPEN** fără opţiunea **FOR INPUT** sau **FOR OUTPUT** realizează ....

k) Instrucţiunea **CLOSEOUT** închide canalul de transfer asociat fişierului de ieşire în limbajul BASIC ....

l) În limbajul BASIC-80, BASIC .... instrucţiunea **CLOSE** fără argumente închide toate fişierele deschise.

m) Un fişier BASIC-PLUS se poate identifica prin .... şi ....

n) Secvenţa de instrucţiuni BASIC–PLUS:

```
10 OPEN "A.DAT" FOR OUTPUT AS
   ASCII & FILE 1 TO WRITE
20 INPUT COD, PREŢ
30 PRINT #2, COD; "," ; PREŢ
40 CLOSE 3
50 END
```

conţine erori în liniile .... , ...

o) Secvenţa de instrucţiuni ABASIC:

```
10 OPEN #1, "A.DAT"
20 WHILE A <> 999
30 INPUT A
40 WRITE #1, A
50 WHILE
60 CLOSE #2
70 END
```

conţine erori în liniile ....

p) Secvenţa de instrucţiuni BASIC-AMSTRAD:

```
10 OPENOUT "A.DAT"
20 INPUT "A, B", A, B
30 WHILE A < > 999
40    PRINT #9, A, B
50    INPUT A, B
60 WEND
70 CLOSE #9
```

conţine erori în liniile ....

r) Secvenţa de instrucţiuni BASIC-80:

```
10 OPEN "O", #1, "AX"
20 INPUT "A, B, C"; A, B, C
30 IF A=999 THEN CLOSE : END
40 PRINT #1, A; ","; B; ","; C
50 GO TO 20
```

poate fi scrisă structurat sub forma:

....

....

....

s) Secvenţa de instrucţiuni BASIC-80:

```
5 DATA 1, 2, 3, 4, 5, 6, 7, 8
10 OPEN "O", #1, "B"
20 FOR I=1 TO 4
30    READ X, Y
40 PRINT #1, X; ","; Y
45 PRINT X, Y
50 NEXT I
60 CLOSE 1
70 END
```

are ca efect ....

☐ Scrieţi cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Să se creeze un fişier dicţionar de organizare secvenţială în limbajele: a) BASIC–AMSTRAD; b) BASIC–COMMODORE; c) BASIC-80; d) BASIC–PLUS; e) ABASIC. Structura fişierului este: cod, denumire.

b) Prestigioasa revistă A.M.C. (Automatică, Management, Calculatoare) ce apare în cadrul Editurii Tehnice a împlinit în anul 1986, 25 ani de la apariţie. Să se alcătuiască un fişier de organizare secvenţială cu toţi autorii care au publicat în revistă. Pe lîngă numele autorilor, fişierul (creat în BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-80, BASIC-PLUS, ABASIC) va mai conţine: titlul articolului, număr AMC, anul apariţiei, număr pagini autor, redactor.

c) Să se creeze un fişier de organizare secvenţială cu coeficienţii şi exponenţii necesari calculului vitezei de aşchiere la strunjire.

d) Să se creeze un fişier de organizare secvenţială, cu zilele onomastice ale celor dragi. Structura fişierului este: Nume, Sex, Ziua de naştere, Telefon, Adresă.

e) Pentru gestionarea sălilor de curs şi seminar dintr-un institut de învăţămînt superior s-a procedat la crearea unui fişier de organizare secvenţială cu următoarea structură: cod-sală; destinaţie (curs/seminar); nume-

cadru didactic ce ocupă sala; denumire disciplină ce se predă în sală; ora (ex.: 14–16); număr-semestru (1/2). Programul de creare se va realiza în limbajele: BASIC–AMSTRAD, BASIC–COMMODORE, BASIC-80, BASIC–PLUS, ABASIC.

☐ **Să se gestioneze pe un suport magnetic** (dischetă/disc) fondul de date privind cantităţile lunare de jucării fabricate de întreprinderea URSULEŢUL. Fişierul se va numi JUCĂRII şi va avea organizare secvenţială. Structura înregistrării este: cod articol, a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3, preţ. Se va proiecta şi realiza un program BASIC.

☐ **Să se gestioneze pe un suport magnetic** (dischetă/disc) fondul de date privind consumurile de materiale (articole) normate şi realizate din ţară şi din import pentru activitatea de reparaţii utilaje grele. Fişierul se va numi CONSUMURI şi va avea organizare secvenţială. Structura înregistrării este: cod articol (material), cod provenienţă (R.S.R./IMPORT), consum normat, consum realizat, preţ de achiziţie. Se va proiecta şi realiza un program BASIC.

SOLUŢIA TEMEI 7

☐ Nu răspundem

☐ Nu răspundem

☐ Nu răspundem

☐ **Programul BASIC este:**

```
 10 REM * Crearea fişier JUCĂRII IN BASIC-80 *
 20 PRINT
 30 OPEN "O", #1, "JUCĂRII.DAT"
 40 PRINT "COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREŢ" ;
 50 PRINT
 60 INPUT COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREŢ
 70 WHILE COD <> 999
 80 WRITE #1, COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREŢ
 90 PRINT
100 PRINT "COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREŢ" ;
110 INPUT COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREŢ
120 WEND
130 CLOSE #1
140 PRINT "Fişierul JUCĂRII.DAT a fost creat"
150 END
```

☐ **Programul BASIC este:**

```
a) 10 REM * Creare fişier CONSUMURI în BASIC-AMSTRAD *
   20 PRINT
   30 OPENOUT "CONSUMURI . DAT"
   40 PRINT "cod, prov, cn, cr, preţ";
```

```
50 INPUT cod, prov, cn, cr, preț
60 WHILE cod < > 999
70 WRITE #9, cod, prov, cn, cr, preț
80 PRINT
90 PRINT "cod, prov, cn, cr, preț";
100 INPUT cod, prov, cn, cr, preț
110 WEND
120 CLOSEOUT
130 PRINT "Fişierul CONSUMURI . DAT a fost creat"
140 END
```

b)
```
10 REM * Creare fişier CONSUMURI IN BASIC-80, GWBASIC *
20 PRINT
30 OPEN "O", # 1, "CONSUMURI.DAT"
40 PRINT "cod, prov, cn, cr, preț"
50 INPUT cod, prov, cn, cr, preț
60 WHILE cod < > 999
70    WRITE # 1, cod, prov, cn, cr, preț
80    PRINT
90    PRINT "cod, prov, cn, cr, preț"
100   INPUT cod, prov, cn, cr, preț
110 WEND
120 CLOSE # 1
130 PRINT "Fişierul CONSUMURI.DAT a fost creat"
140 END
```

CONVERSAȚIA 8

Proiectarea și realizarea unui program BASIC ghidat de un meniu pentru actualizarea, consultarea și listarea unui fișier secvențial de livrări. Instrucțiunile INKEY$, GET, ERASE, OPENIN, CLOSEIN, WRITE# și ON... GOTO. Funcția EOF. Fișiere cu acces direct. Depanarea unui program. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLELE 8 – PC, m, M, F

☐ De la problemă la program

În cadrul acestei conversaţii vom "umbla" în fişierul LIVRĂRI.DAT (creat în conversaţia precedentă) în ideea familiarizării dvs. cu instrucţiunile BASIC pentru prelucrarea fişierelor de organizare secvenţială.

Vom proiecta şi realiza un program ghidat de un *meniu* care să ofere posibilitatea optării pentru una din următoarele (şase) funcţiuni care se afişează efectiv pe ecran:

1) Încarcă fişier LIVRĂRI.DAT
2) Salvează fişier
3) Căutare în fişier
4) Actualizare fişier
5) Listare fişier
6) End

Fiecare opţiune va corespunde unei subrutine care realizează funcţiunea afişată.

Singurul „efort" care se cere utilizatorului acestui program este acela de a tasta cifra corespunzătoare opţiunii dorite, fără a-l mai „chinui" de-a acţiona imediat tasta [CR]. Pentru reîntoarcerea în meniu se va tasta de fiecare dată [CR].

Pentru fiecare din cele şase funcţiuni ale programului se vor afişa mesaje de avertizare, corespunzătoare.

Dacă, de exemplu, utilizatorul va opta pentru încărcarea fişierului LIVRĂRI·DAT de pe dischetă în memoria calculatorului (opţiune obligatorie la începutul execuţiei programului) programul (subrutina) va afişa mesajul „Fişier încărcat".

Dacă utilizatorul va opta pentru funcţiunea „căutare în fişier", programul (subrutina) va afişa un mesaj prin care să se solicite codul beneficiarului. În urma căutării secvenţiale a beneficiarului cu codul indicat în fişierul LIVRĂRI.DAT se va preciza dacă acesta a fost sau nu găsit.

Cît priveşte actualizarea fişierului (v. opţiunea 4) se va modifica valoarea cantităţii de benzină livrate. În funcţie de codul beneficiarului se va actualiza cîmpul corespunzător (CANT) cu o nouă valoare. În caz de eroare (beneficiarul nu se află în fişier) se va afişa un mesaj explicativ.

Pentru listarea fişierului (înainte de actualizare, sau după actualizare) se va scrie o subrutină care să corespundă opţiunii 5.

Dacă utilizatorul a ales funcţiunea 2, programul (subrutina) va salva pe suportul magnetic respectiv ultima versiune a fişierului LIVRĂRI.DAT. În urma salvării se va afişa mesajul "salvat fişier".

Pentru oprirea execuţiei programului se va opta pentru funcţiunea 6 căreia i se va prevedea, de asemenea, în cadrul programului o subrutină.

☐ Specificaţii de programare

Schema de sistem, formatul datelor de ieşire, structura înregistrării de ieşire, tabela de variabile şi specificaţiile de programare sînt ilustrate în modulul de analiză structurată, fig. 8.1.

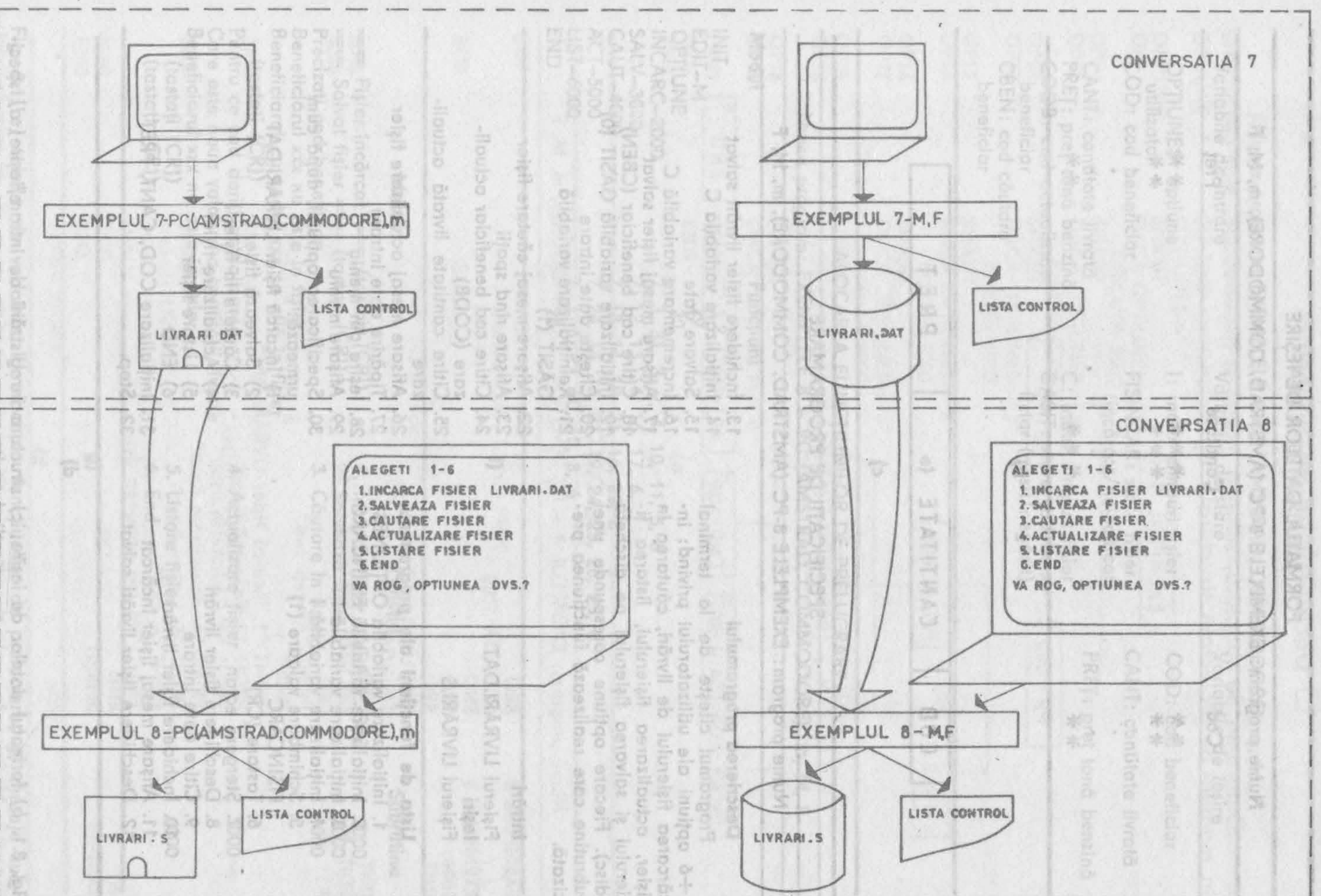
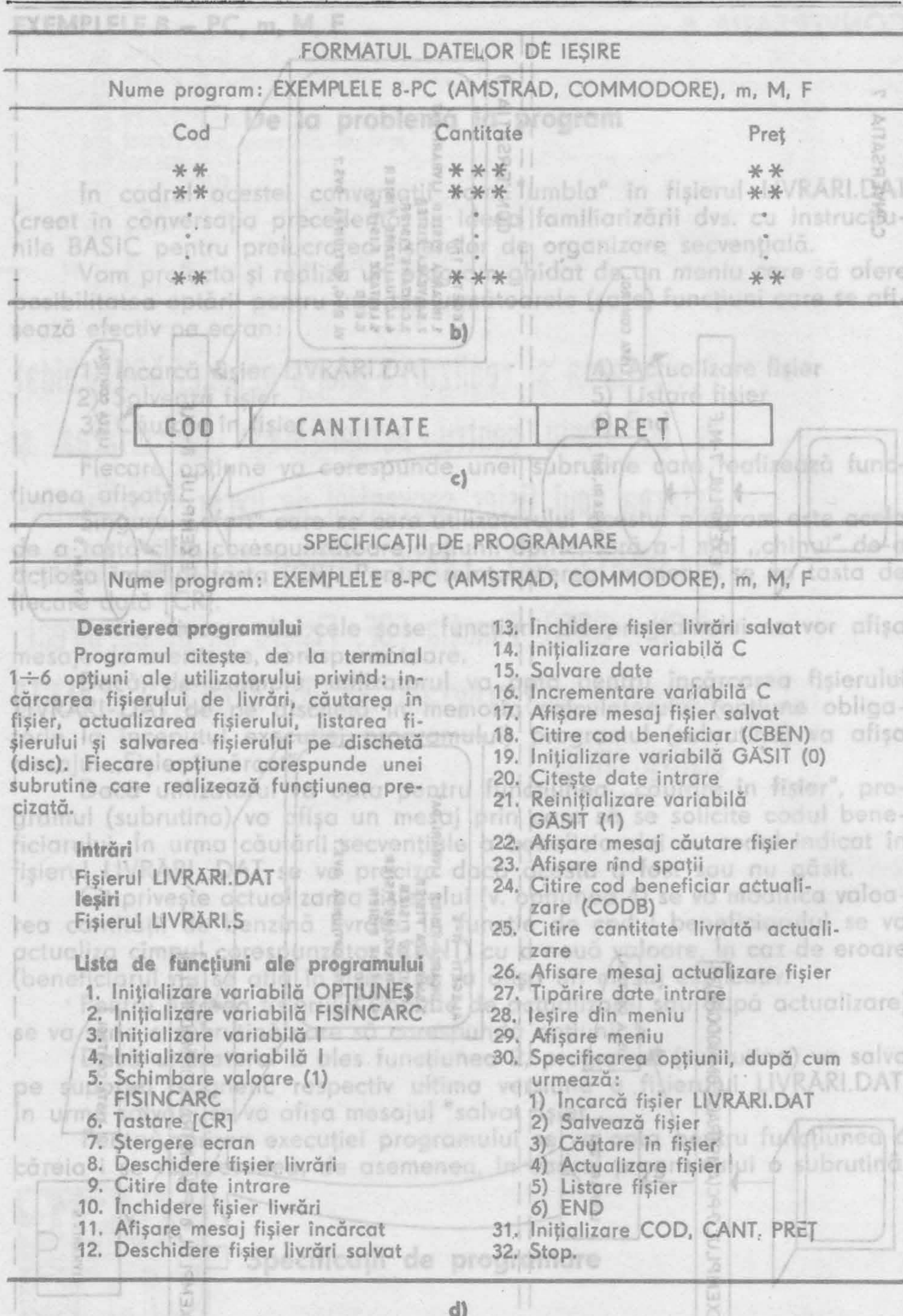


Fig. 8.1. Modulul de analiză structurată: *a)* schema de sistem.



FORMATUL DATELOR DE IEŞIRE

Nume program: EXEMPLELE 8-PC (AMSTRAD, COMMODORE), m, M, F

Cod	Cantitate	Preţ
**	***	**
**	***	**
.	.	.
.	.	.
**	***	**

b)

COD	CANTITATE	PREŢ

c)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 8-PC (AMSTRAD, COMMODORE), m, M, F

Descrierea programului

Programul citeşte de la terminal 1÷6 opţiuni ale utilizatorului privind: încărcarea fişierului de livrări, căutarea în fişier, actualizarea fişierului, listarea fişierului şi salvarea fişierului pe dischetă (disc). Fiecare opţiune corespunde unei subrutine care realizează funcţiunea precizată.

Intrări

Fişierul LIVRĂRI.DAT

Ieşiri

Fişierul LIVRĂRI.S

Lista de funcţiuni ale programului

1. Iniţializare variabilă OPŢIUNE$
2. Iniţializare variabilă FISINCARC
3. Iniţializare variabilă I
4. Iniţializare variabilă I
5. Schimbare valoare (1) FISINCARC
6. Tastare [CR]
7. Ştergere ecran
8. Deschidere fişier livrări
9. Citire date intrare
10. Închidere fişier livrări
11. Afişare mesaj fişier încărcat
12. Deschidere fişier livrări salvat
13. Închidere fişier livrări salvat
14. Iniţializare variabilă C
15. Salvare date
16. Incrementare variabilă C
17. Afişare mesaj fişier salvat
18. Citire cod beneficiar (CBEN)
19. Iniţializare variabilă GĂSIT (0)
20. Citeşte date intrare
21. Reiniţializare variabilă GĂSIT (1)
22. Afişare mesaj căutare fişier
23. Afişare rînd spaţii
24. Citire cod beneficiar actualizare (CODB)
25. Citire cantitate livrată actualizare
26. Afişare mesaj actualizare fişier
27. Tipărire date intrare
28. Ieşire din meniu
29. Afişare meniu
30. Specificarea opţiunii, după cum urmează:
    1) Încarcă fişier LIVRĂRI.DAT
    2) Salvează fişier
    3) Căutare în fişier
    4) Actualizare fişier
    5) Listare fişier
    6) END
31. Iniţializare COD, CANT. PREŢ
32. Stop.

d)

Fig. 8.1. *b)* formatul datelor de ieşire; *c)* structura înregistrării de intrare/ieşire; *d)* specificaţii de programare;

TABELA DE VARIABILE		
Nume program: EXEMPLELE 8-PC (AMSTRAD, COMMODORE), m, M, F		
Variabile de intrare	Variabile de stare	Variabile de ieșire
OPȚIUNE$: opțiune utilizator	I: index cîmpuri fișier	COD: cod beneficiar
COD: cod beneficiar	FISINCAR: stare fișier (încărcat/neîncărcat)	CANT: cantitate livrată
CANT: cantitate livrată		PREȚ: preț tonă benzină
PREȚ: preț tonă benzină	C: Index cîmpuri fișier	
CODB: cod actualizare beneficiar	GĂSIT: stare cod beneficiar (găsit/negăsit)	
CBEN: cod căutare beneficiar		

e)

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE	
Nume program: EXEMPLELE 8 – PC (AMSTRAD, COMMODORE), m, M, F	
Modul	**Funcțiuni**
INIT	1,2
EDIT–M	29
OPȚIUNE	30
INCARC–2000	7, 3, 31, 8, 9, 4, 10, 11, 5, 6
SALV–3000	7, 12, 13, 14, 16, 17, 6, 5
CAUT–4000	7, 18, 14, 19, 21, 16, 22, 6
ACT–5000	7, 24, 14, 16, 21, 19, 24, 25, 26, 23, 6
LIST–6000	7, 3, 4, 27, 20, 10, 8, 6
END	28

f)

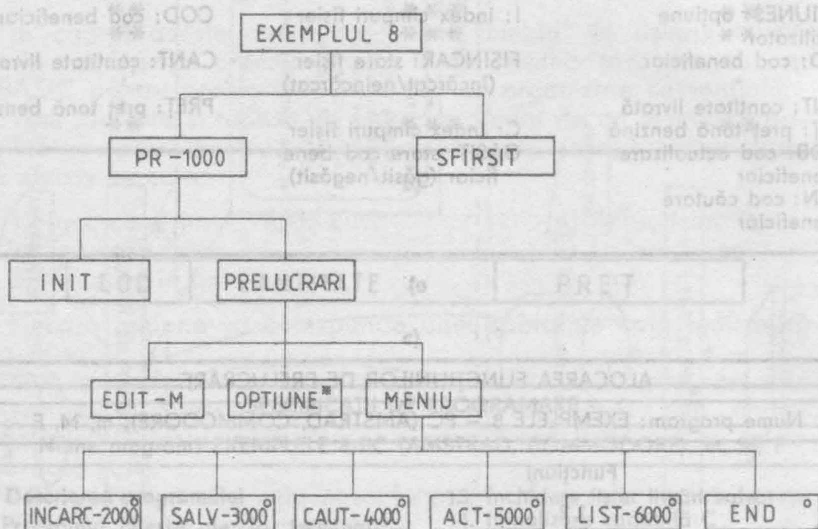
Mesaj	Funcție	Subrutina
== Fișier încărcat == (tastați [CR])	1. Încarcă fișier LIVRARI . DAT	2000
== Salvat fișier == (tastați [CR])	2. Salvează fișier	3000
Precizați cod beneficiar Beneficiarul xxx nu este în fișier Beneficiarul cu codul xxx se află în fișier (tastați [CR])	3. Căutare în fișier	4000
Pentru ce cod doriți actualizarea Care este noua valoare a cantității livrate Beneficiarul xxx nu este în fișier	4. Actualizare fișier	5000
(tastați [CR])	5. Listare fișier	6000
(tastați [CR])	6. End	

g)

Fig. 8.1. *e)* tabela de variabile; *f)* alocarea funcțiunilor de prelucrare; *g)* mesaje subrutine.

□ Documentaţia de proiectare

Diagrama de structură, schema logică şi pseudocodul sînt prezentate în modulul de proiectare structurată, fig. 8.2.



a)

PSEUDOCODUL

Nume program: EXEMPLUL 8 – PC (AMSTRAD, COMMODORE), m, M, F

```
EXEMPLUL 8     SEQ
               EXECUTĂ 1000
EXEMPLUL 8     END
1000           SEQ
INIT           SEQ
               OPŢIUNE$=""
               FISINCARC=0
INIT           END
EDIT–M         CIT TIMP OPŢIUNE$ ≠ 6
A1             SEQ
               PRINT "Alegeţi 1–6"
               PRINT
               PRINT "1. Încarcă fişier LIVRĂRI.DAT"
               PRINT "2. Salvează fişier"
               PRINT "3. Căutare în fişier"
               PRINT "4. Actualizare fişier"
               PRINT "5. Listare fişier"
               PRINT "6. End"
               PRINT "Vă rog opţiunea dvs."
A1             END
EDIT–M         SFIRŞIT
```

b)

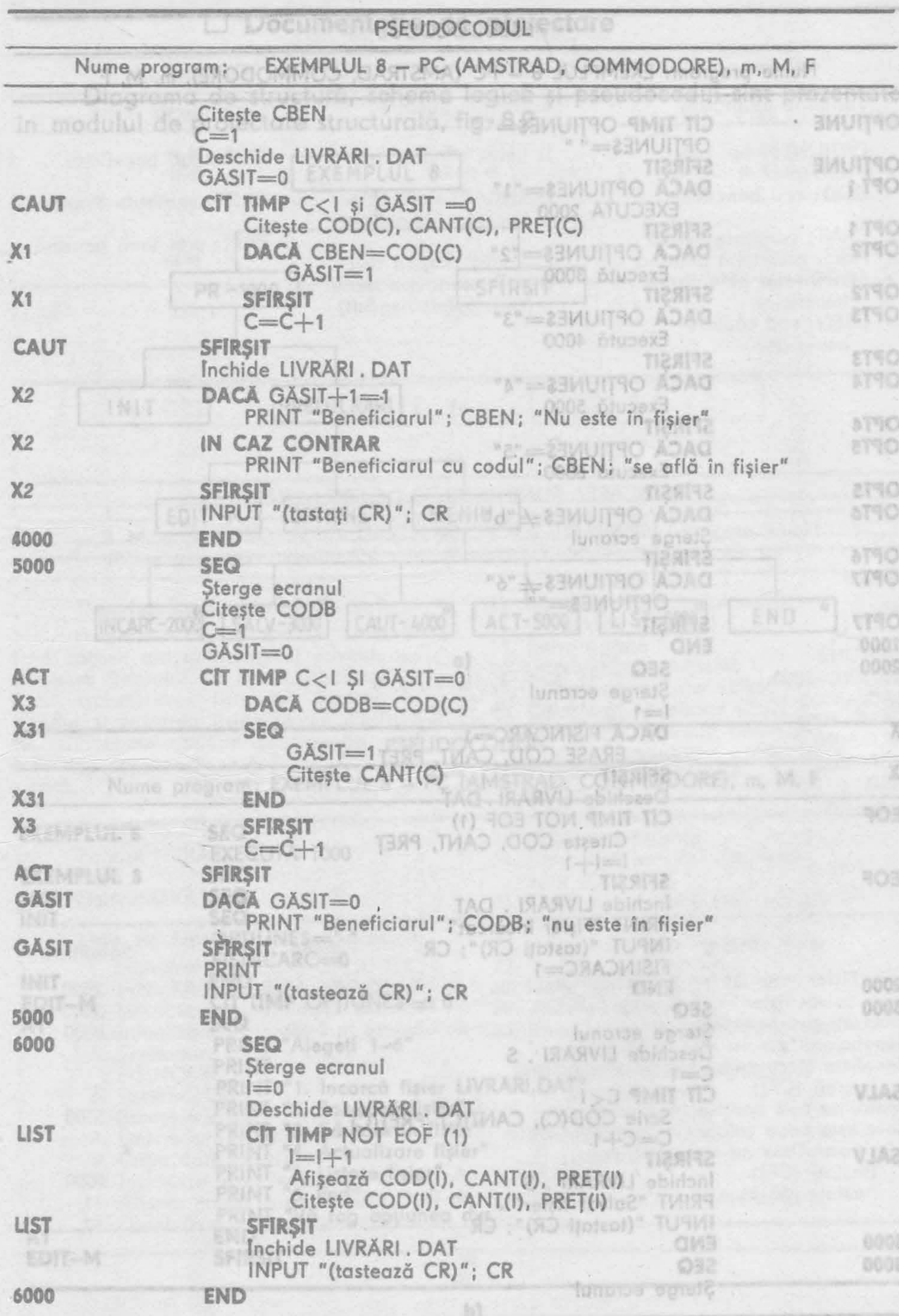
Fig. 8.2. Modulul de proiectare structurată: *a)* diagrama de structură; *b)* pseudocodul;

PSEUDOCODUL

Nume program: EXEMPLUL 8 – PC (AMSTRAD, COMMODORE), m, M, F

```
OPŢIUNE     CÎT TIMP OPŢIUNE$=" "
              OPŢIUNE$=" "
OPŢIUNE     SFÎRŞIT
OPT 1       DACĂ OPŢIUNE$="1"
                EXECUTĂ 2000
OPT 1       SFÎRŞIT
OPT2        DACĂ OPŢIUNE$="2"
                Execută 3000
OPT2        SFÎRŞIT
OPT3        DACĂ OPŢIUNE$="3"
                Execută 4000
OPT3        SFÎRŞIT
OPT4        DACĂ OPŢIUNE$="4"
                Execută 5000
OPT4        SFÎRŞIT
OPT5        DACĂ OPŢIUNE$="5"
                Execută 6000
OPT5        SFÎRŞIT
OPT6        DACĂ OPŢIUNE$≠"b"
            Şterge ecranul
OPT6        SFÎRŞIT
OPT7        DACĂ OPŢIUNE$≠"6"
              OPŢIUNE$=""
OPT7        SFÎRŞIT
1000        END
2000            SEQ
                Şterge ecranul
                I=1
X               DACĂ FISINCARC=1
                    ERASE COD, CANT, PREŢ
X               SFÎRŞIT
                Deschide LIVRĂRI . DAT
EOF             CÎT TIMP NOT EOF (1)
                    Citeşte COD, CANT, PREŢ
                    I=I+1
EOF             SFÎRŞIT
                Închide LIVRĂRI . DAT
                PRINT "Fişier încărcat"
                INPUT "(tastaţi CR)"; CR
                FISINCARC=1
2000            END
3000        SEQ
            Şterge ecranul
            Deschide LIVRĂRI . S
            C=1
SALV        CÎT TIMP C<I
                Scrie COD(C), CANT(C), PREŢ(C)
                C=C+1
SALV        SFÎRŞIT
            Închide LIVRĂRI . S
            PRINT "Salvat fişier"
            INPUT "(tastaţi CR)"; CR
3000        END
4000        SEQ
            Şterge ecranul
```

Fig. 8.2. *b*



PSEUDOCODUL

Nume program: EXEMPLUL 8 – PC (AMSTRAD, COMMODORE), m, M, F

```
          Citeşte CBEN
          C=1
          Deschide LIVRĂRI . DAT
          GĂSIT=0
CAUT      CÎT TIMP C<I şi GĂSIT =0
              Citeşte COD(C), CANT(C), PREŢ(C)
X1            DACĂ CBEN=COD(C)
                  GĂSIT=1
X1            SFÎRŞIT
              C=C+1
CAUT      SFÎRŞIT
          Închide LIVRĂRI . DAT
X2        DACĂ GĂSIT+1=1
              PRINT "Beneficiarul"; CBEN; "Nu este în fişier"
X2        IN CAZ CONTRAR
              PRINT "Beneficiarul cu codul"; CBEN; "se află în fişier"
X2        SFÎRŞIT
          INPUT "(tastaţi CR)"; CR
4000      END
5000      SEQ
          Şterge ecranul
          Citeşte CODB
          C=1
          GĂSIT=0
ACT       CÎT TIMP C<I ŞI GĂSIT=0
X3            DACĂ CODB=COD(C)
X31           SEQ
                  GĂSIT=1
                  Citeşte CANT(C)
X31           END
X3            SFÎRŞIT
              C=C+1
ACT       SFÎRŞIT
GĂSIT     DACĂ GĂSIT=0
              PRINT "Beneficiarul"; CODB; "nu este în fişier"
GĂSIT     SFÎRŞIT
          PRINT
          INPUT "(tastează CR)"; CR
5000      END
6000          SEQ
              Şterge ecranul
              I=0
              Deschide LIVRĂRI . DAT
LIST          CÎT TIMP NOT EOF (1)
                  I=I+1
                  Afişează COD(I), CANT(I), PREŢ(I)
                  Citeşte COD(I), CANT(I), PREŢ(I)
LIST              SFÎRŞIT
              Închide LIVRĂRI . DAT
              INPUT "(tastează CR)"; CR
6000      END
```

Fig. 8.2. *b*

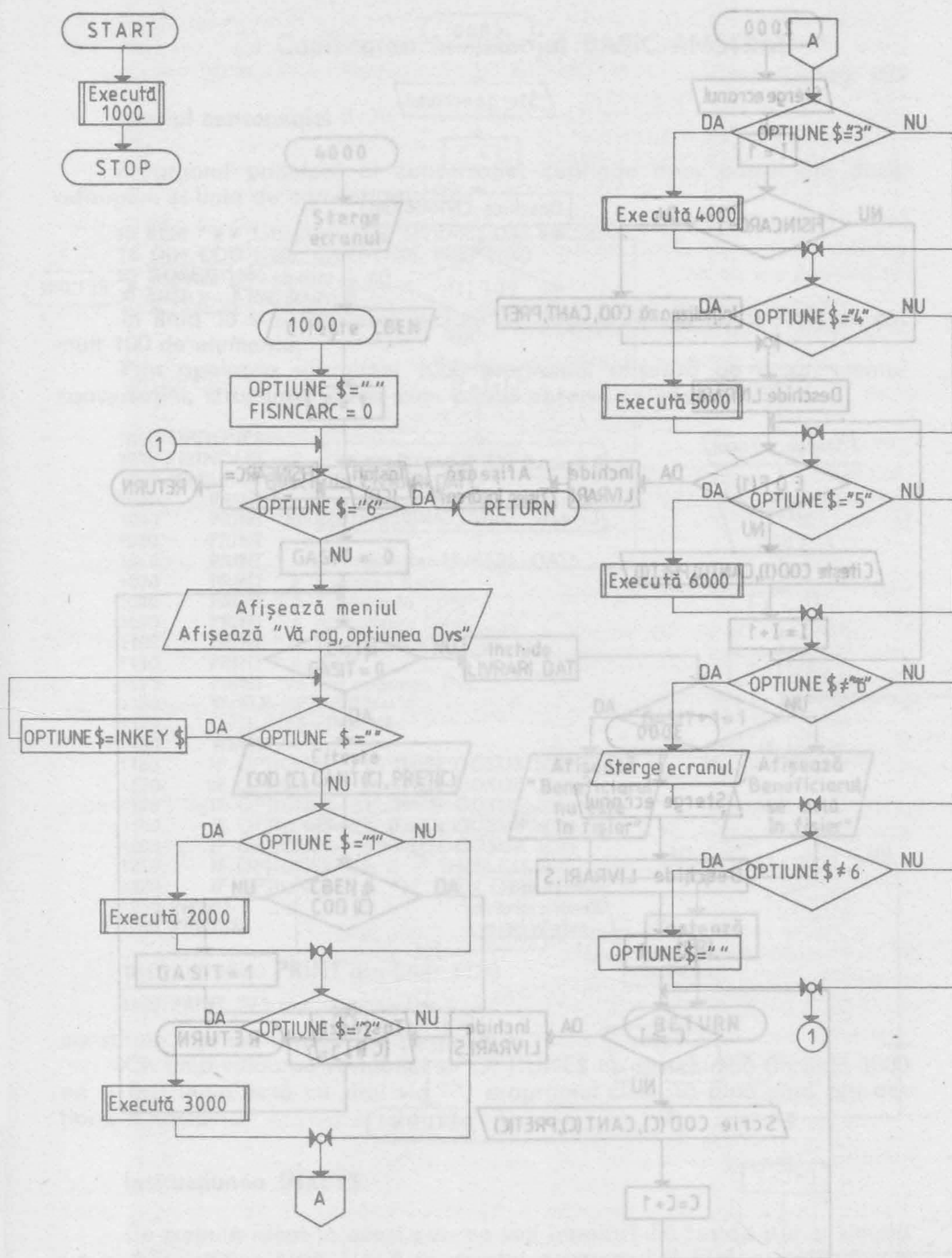


Fig. 8.2. c) schema logică.

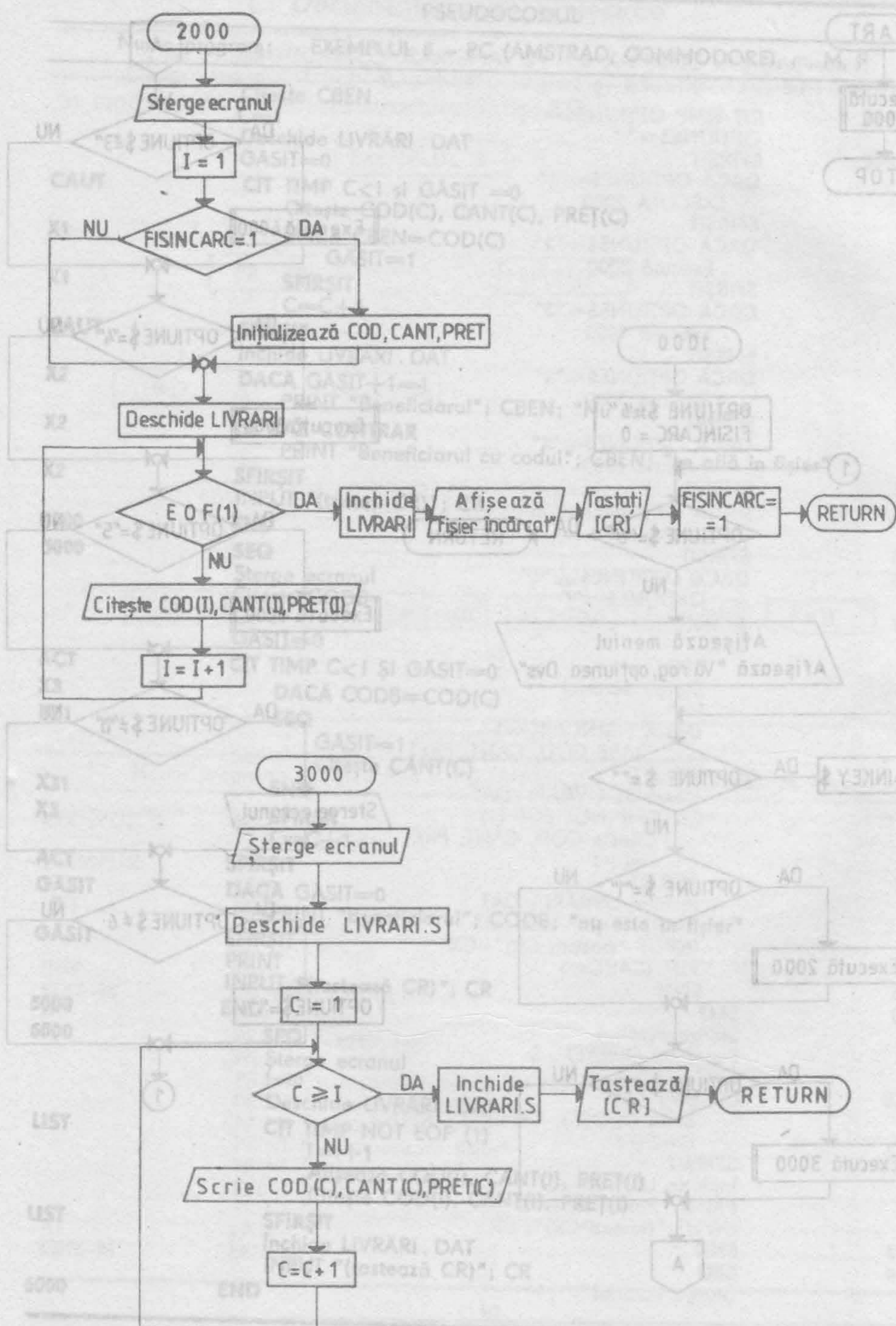

2000
Sterge ecranul
I = 1
NU
FISINCARC=1
DA
Iniţializează COD, CANT, PRET
Deschide LIVRARI
E O F(1)
DA
Inchide LIVRARI
Afişează "Fişier încărcat"
Tastaţi [CR]
FISINCARC= =1
RETURN
NU
Citeşte COD(I),CANT(I),PRET(I)
I = I+1
3000
Sterge ecranul
Deschide LIVRARI.S
C = 1
C ≥ I
DA
Inchide LIVRARI.S
Tastează [CR]
RETURN
NU
Scrie COD(C),CANT(C),PRET(C)
C=C+1

Fig. 8.2. c

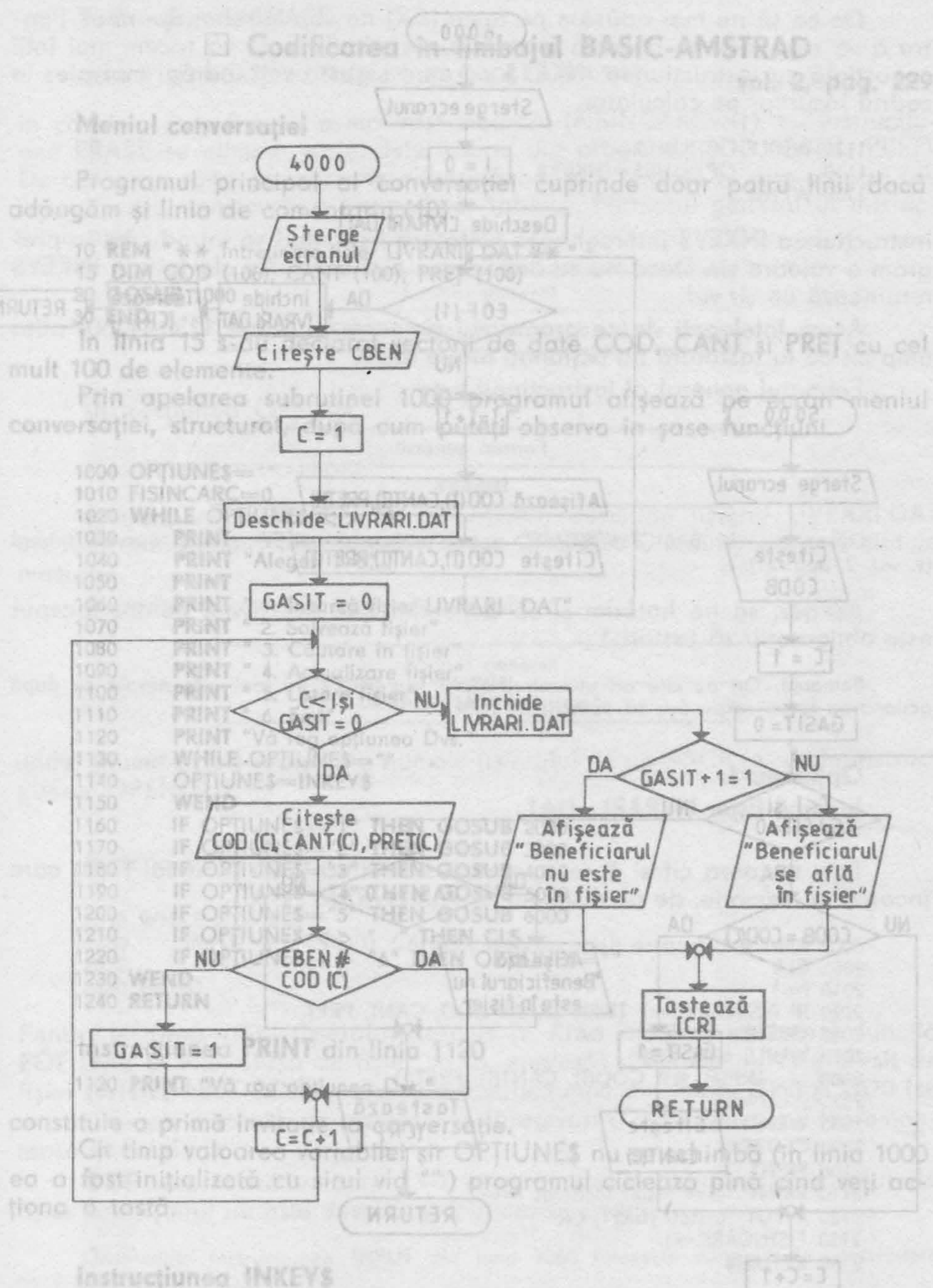

4000
Şterge ecranul
Citeşte CBEN
C = 1
Deschide LIVRARI.DAT
GASIT = 0
C<I şi GASIT = 0
NU
DA
Inchide LIVRARI.DAT
Citeşte COD (C), CANT(C), PRET(C)
CBEN # COD (C)
NU
DA
GASIT=1
C=C+1
DA
GASIT+1=1
NU
Afişează "Beneficiarul nu este în fişier"
Afişează "Beneficiarul se află în fişier"
Tastează [CR]
RETURN

Fig. 8.2. c

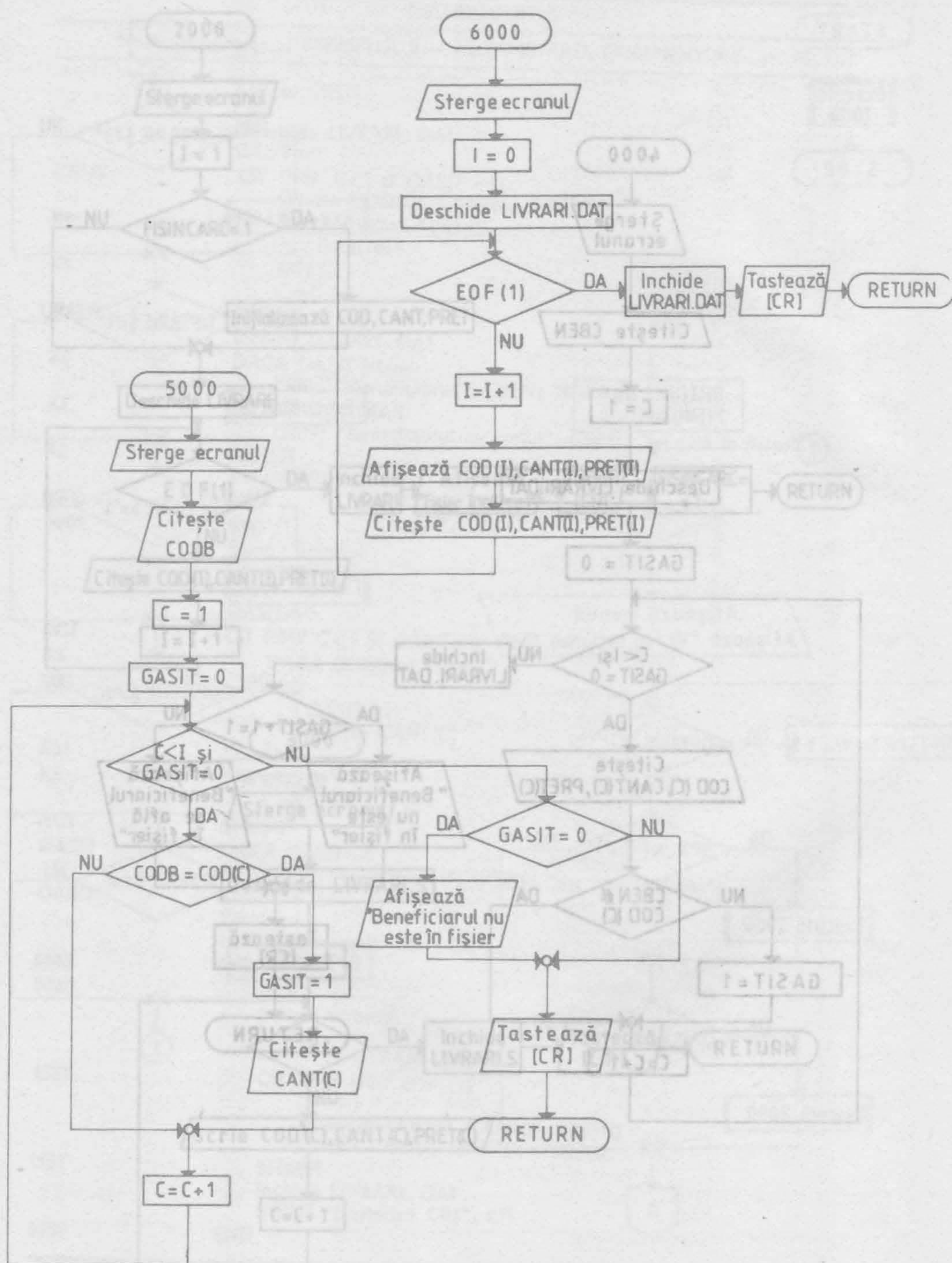

6000
Sterge ecranul
I = 0
Deschide LIVRARI.DAT
EOF (1)
DA
Inchide LIVRARI.DAT
Tastează [CR]
RETURN
NU
I=I+1
Afişează COD(I),CANT(I),PRET(I)
Citeşte COD(I),CANT(I),PRET(I)
5000
Sterge ecranul
Citeşte CODB
C = 1
GASIT= 0
C<I şi GASIT=0
NU
DA
CODB = COD(C)
NU
DA
GASIT = 1
Citeşte CANT(C)
C=C+1
GASIT= 0
DA
NU
Afişează "Beneficiarul nu este în fişier
Tastează [CR]
RETURN

Fig. 8.2. c

☐ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 229

Meniul conversaţiei

Programul principal al conversaţiei cuprinde doar patru linii dacă adăugăm şi linia de comentariu (10).

```
10 REM "** Întreţinere fişier LIVRĂRI . DAT ** "
15 DIM COD (100), CANT (100), PREŢ (100)
20 GOSUB 1000
30 END
```

În linia 15 s-au declarat vectorii de date COD, CANT şi PREŢ cu cel mult 100 de elemente.

Prin apelarea subrutinei 1000 programul afişează pe ecran meniul conversaţiei, structurat, după cum puteţi observa în şase funcţiuni.

```
1000 OPŢIUNE$=""
1010 FISINCARC=0
1020 WHILE OPŢIUNE$ < > "6"
1030     PRINT
1040     PRINT "Alegeţí 1–6"
1050     PRINT
1060     PRINT " 1. Încarcă fişier LIVRĂRI . DAT"
1070     PRINT " 2. Salvează fişier"
1080     PRINT " 3. Căutare în fişier"
1090     PRINT " 4. Actualizare fişier"
1100     PRINT " 5. Listare fişier"
1110     PRINT " 6. End"
1120     PRINT "Vă rog opţiunea Dvs."
1130     WHILE OPŢIUNE$=""
1140     OPŢIUNE$=INKEY$
1150     WEND
1160     IF OPŢIUNE$="1" THEN GOSUB 2000
1170     IF OPŢIUNE$="2" THEN GOSUB 3000
1180     IF OPŢIUNE$="3" THEN GOSUB 4000
1190     IF OPŢIUNE$="4" THEN GOSUB 5000
1200     IF OPŢIUNE$="5" THEN GOSUB 6000
1210     IF OPŢIUNE$ <> " " THEN CLS
1220     IF OPŢIUNE$ < > "6" THEN OPŢIUNE$=""
1230 WEND
1240 RETURN
```

Instrucţiunea **PRINT** din linia 1120

```
1120 PRINT "Vă rog opţiunea Dvs."
```

constituie o primă invitaţie la conversaţie.

Cît timp valoarea variabilei şir OPŢIUNE$ nu se schimbă (în linia 1000 ea a fost iniţializată cu şirul vid "") programul ciclează pînă cînd veţi acţiona o tastă.

Instrucţiunea INKEY$

Ce trebuie făcut în acest caz, ne veţi întreba? Să tastaţi pur şi simplu un număr cuprins între 1 şi 6 (v. meniul programului) fără a mai acţiona însă tasta [CR] după cum sînteţi obişnuiţi.

De ce să nu mai apăsăm pe tasta [CR] ne veţi întreba, din nou? Pentru a vă răspunde la această întrebare, permiteţi-ne să vă facem mai întîi cunoştinţă cu instrucţiunea **INKEY$** pe care sigur o veţi îndrăgi mai ales în cadrul jocurilor pe calculator.

```
1130 WHILE OPŢIUNE$=""
1140       OPŢIUNE$=INKEY$
1150 WEND
```

Instrucţiunea **INKEY$** interoghează tastatura aşteptînd să introduceţi în program o valoare şir. Dacă nu se acţionează nici o tastă a claviaturii, **INKEY$** returnează un şir vid.

Acum înţelegeţi de ce programul ciclează (v. bucla 1130–1150) atîta timp cît de la tastatură nu acţionaţi nici o tastă.

Formatul general al instrucţiunii este

Format general
INKEY$

Observaţie. In BASIC–AMSTRAD puteţi utiliza şi **INKEY** pentru numere întregi (v. vol. 2, pag. 31).

Aşadar, să ne hotărîm şi să acţionăm una din taste. Pentru început este obligatoriu să tastaţi 1.

Remarcă. Ori de cîte ori utilizaţi **INKEY$** sau **INKEY** nu mai este nevoie ca după acţionarea tastei respective să apăsaţi pe [**CR**].

Opţiunea 1.
Incarcă fişier LIVRĂRI . DAT

Prin tastarea cifrei 1 se apelează subrutina 2000 (v. linia 1160) care încarcă în memorie, de pe dischetă fişierul LIVRĂRI . DAT.

```
2000 REM * 1. Incarcă fişier LIVRĂRI . DAT
2005 CLS
2010 I=1
2020 IF FISINCARC=1 THEN ERASE COD, CANT, PREŢ
2040 OPENIN "LIVRĂRI . DAT"
2050 WHILE NOT EOF
2060      INPUT #9, COD(I), CANT(I), PREŢ(I)
2070      I=I+1
2080 WEND
2090 CLOSEIN
2100 PRINT
2110 PRINT "== Fişier încărcat =="
2120 INPUT "tastaţi [CR]"; CR
2130 FISINCARC=1
2140 RETURN
```

Pentru încărcarea prin program a fişierului trebuie să ne asigurăm dacă fişierul a mai fost sau nu încărcat. După aceea vom deschide fişierul pe un canal de transfer, vom citi cîte o înregistrare iar în final vom închide canalul de transfer asociat fişierului.

Instrucţiunea ERASE

```
2020 IF FISINCARC=1 THEN ERASE COD, CANT, PREŢ
```

În cazul în care fişierul a mai fost încărcat (FISINCARC=1), cu instrucţiunea **ERASE** se elimină variabilele tablou din program (COD, CANT, PREŢ). De notat că, zona de memorie eliberată poate fi folosită în alte scopuri (de exemplu redimensionarea variabilelor tablou). Formatul general al instrucţiunii este:

Format general
ERASE ⟨listă variabile tablou⟩

Instrucţiunea OPENIN

```
2040 OPENIN "LIVRĂRI.DAT"
```

Instrucţiunea **OPENIN** (v. linia 2040) deschide fişierul LIVRĂRI.DAT creat în conversaţia precedentă (7) cu **OPENOUT**, înainte de a fi citit în memorie.

Formatul general al instrucţiunii este:

Format general
OPENIN "⟨nume-fişier⟩"

unde, ⟨nume-fişier⟩ reprezintă numele fişierului de tip ASCII folosit în instrucţiunea **OPENOUT**.

Funcţia EOF

```
2050 WHILE NOT EOF
2060      INPUT #9, COD(I), CANT(I), PREŢ(I)
2070      I=I+1
2080 WEND
```

Pentru testarea stării fişierului deschis (v. linia 2050) se utilizează funcţia **EOF** (**E**nd **o**f **F**ile, sfîrşit de fişier, în lb. engleză). Cît timp nu este sfîrşit de fişier **(WHILE NOT EOF)** se execută instrucţiunile din liniile 2060 şi 2070 (se citeşte de pe dischetă, în memorie, o înregistrare din fişier şi se incrementează variabila I).

EOF returnează valoarea −1 atunci cînd se detectează sfîrşitul de fişier sau fişierul nu este deschis şi 0 în caz contrar.

Observaţie. Instrucţiunea **INPUT** din linia 2060 foloseşte canalul de deschidere nr. 9 (discheta).

Instrucţiunea CLOSEIN

```
2090 CLOSEIN
```

Pentru închiderea fişierului de livrări deschis cu **OPENIN** limbajul BASIC AMSTRAD utilizează instrucţiunea **CLOSEIN.**

Formatul general al instrucţiunii este

Format general
CLOSEIN

Observaţie. Pentru reintoarcerea in meniu acţionaţi tasta [**CR**].

Opţiunea 2.
Salvează fişier

Prin tastarea cifrei 2 se apelează subrutina 3000 (v. linia 1170) care salvează pe dischetă fişierul de date existent în memorie.

Altfel spus, subrutina creează în **OUTPUT**, pe dischetă un fişier secvenţial ASCII.

```
3000 REM * 2. Salvează fişier
3005 CLS
2010 OPENOUT "LIVRĂRI.S"
3020 C=1
3030 WHILE C < I
3040     WRITE #9, COD(C), CANT(C), PREŢ(C)
3050     C=C+1
3060 WEND
3070 CLOSEOUT
3080 PRINT "Salvat fişier=="
3090 INPUT "tastaţi [CR]"; CR
3100 RETURN
```

Observaţie. Pentru reintoarcerea in meniu tastaţi [**CR**].

Opţiunea 3.
Căutare în fişier

Prin tastarea cifrei 3 se apelează subrutina 4000 (v. linia 1180) care verifică dacă un beneficiar se găseşte sau nu în fişierul de livrări.

```
4000 REM * 3. Căutare în fişier
4010 CLS
4020 INPUT "Precizaţi cod beneficiar"; CBEN
4030 C=1
4035 OPENIN "LIVRĂRI.DAT"
4040 GĂSIT=0
4050 WHILE C<I AND GĂSIT=0
4052     INPUT #9, COD(C), CANT(C), PREŢ(C)
4054     IF CBEN <> COD(C) THEN 4060
4056     GĂSIT=1
4060     C=C+1
4070     WEND
4070 WEND
4072 CLOSEIN
4075 ON GĂSIT+1 GOTO 4080, 4092
4078 REM Beneficiarul a fost sau nu găsit în fişier.
4080 PRINT "Beneficiarul"; CBEN; "nu este în fişier"
```

```
4090 PRINT
4091 GOTO 4100
4092 PRINT "Beneficiarul cu codul"; CBEN; "se află în fişier"
4100 INPUT "(tastaţi [CR])"; CR
4110 RETURN
```

Regăsirea beneficiarului se face în funcţie de codul său (v. linia 4020). Cît timp C<I şi GĂSIT=0 (condiţie compusă!) se execută instrucţiunile din liniile 4052–4060. În urma identificării beneficiarului variabila GĂSIT se poziţionează pe 1.

La ieşirea din buclă se închide fişierul şi se imprimă, funcţie de valoarea variabilei de stare GĂSIT, mesajul privind rezultatul cercetării secvenţiale a fişierului de livrări: "Beneficiarul a fost sau nu găsit în fişier" (v. liniile 4080 şi 4092).

Instrucţiunea ON . . . GOTO

```
4075 ON GĂSIT+1 GOTO 4080, 4092
```

Cu instrucţiunea **ON . . . GOTO** (v. linia 4075) se realizează saltul la una din liniile de după **GOTO**: 4080/4092.

Linia 4080 se selectează atunci cînd GĂSIT+1 are valoarea 1. Pentru valoarea 2 a aceleiaşi variabile se selectează linia 4092.

Formatul general al instrucţiunii este

Format general
ON ⟨expresie⟩ **GOTO** ⟨listă-nr. linii⟩

în care: ⟨expresie⟩ este un număr întreg cuprins între 0 şi 255; ⟨lista – nr. linii⟩ reprezintă numerele de linie ale programului, selectate în funcţie de valoarea ⟨expresie⟩.

Observaţii

- Dacă ⟨expresie⟩ este egală cu zero sau mai mare decît numărul de elemente din listă, selecţia nu va mai avea loc;
- Dacă ⟨expresie⟩ este negativă apare un mesaj de eroare.

Opţiunea 4.
Actualizare fişier

Prin tastarea cifrei 4 se apelează subrutina 5000 (v. linia 1190) care actualizează fişierul de livrări (se modifică valoarea cîmpului CANT).

```
5000 REM * 4. Actualizare fişier
5005 CLS
5010 INPUT "Pentru ce cod doriţi actualizarea"; CODB
5020 C=1
5030 GĂSIT=0
5040 WHILE C<I AND GĂSIT=0
5050      IF CODB <> COD(C) THEN 5060
5051      INPUT "Care este noua valoare a cantităţii livrate"; CANT(C)
5052      GĂSIT=1
5060      C=C+1
5070 WEND
```

```
5080 IF GĂSIT=0 THEN PRINT "Beneficiarul"; CODB; "nu este în fişier"
5090 PRINT
5100 INPUT "(tastaţi[CR])"; CR
5110 RETURN
```

Actualizarea cîmpului CANT are loc atunci cînd CODB (v. linia 5010) este egal cu COD(C) (v. linia 5050). Prin program (v. linia 5051) se citeşte dinamic CANT(C), adică noua valoare a cantităţii livrate. Din acel moment variabila de stare GĂSIT se poziţionează pe 1. În situaţia cînd beneficiarul nu se află în fişier, instrucţiunea din linia 5080 afişează un mesaj explicit.

Observaţie. Pentru reîntoarcerea în meniu acţionaţi tasta [**CR**].

Opţiunea 5.
Listare fişier

Prin tastarea cifrei 5 se apelează subrutina 6000 (v. linia 1200) care listează conţinutul fişierului de livrări.

```
6000 REM * 5. Listare fişier
6010 CLS
6015 PRINT "Cod Cantitate Preţ"
6020 I=0
6030 OPENIN "LIVRĂRI.DAT"
6050 WHILE NOT EOF
6055     I=I+1
6060     PRINT COD(I); " "; CANT(I); " "; PREŢ(I)
6070     INPUT #9, COD(I), CANT(I), PREŢ(I)
6080 WEND
6090 CLOSEIN
6100 INPUT "(tastaţi [CR])"; CR
6110 RETURN
```

Observaţie. Pentru reîntoarcerea în meniu acţionaţi tasta [**CR**].

Opţiunea 6. End

Prin acţionarea tastei cu cifra 6 se revine la linia 30 din programul principal. În acel moment conversaţia s-a încheiat.

Aplicaţie. Să se proiecteze şi realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea şi listarea unei fişier privind cheltuielile de întreţinere, nominalizate ale locatarilor unui bloc. Se urmăreşte ca la sfîrşitul fiecărei luni să se genereze un raport identic cu cel pe care-l lecturaţi cu o mare atenţie în holul blocului dvs. Pentru stabilirea informaţiilor minimale necesare constituirii fondului de date vă recomandăm să-l abordaţi direct pe ... administrator.

□ Codificarea în limbajul BASIC-COMMODORE

Diferenţele de scriere a programului BASIC-COMMODORE apar la instrucţiunile **WHILE–WEND** (BASIC-COMMODORE nu are implementată structura **CÎT TIMP**), **OPEN, CLOSE, INPUT #, WRITE #** şi **INKEY** (v. instrucţiunea **GET**).

Cum dvs. aveţi deja experienţă în portabilitatea programelor, considerăm că nu vă va fi greu să scrieţi singuri programul pentru calculatorul COMMODORE operînd în programul precedent următoarele modificări:

- Buclele **WHILE–WEND** din liniile 1020, 1230, 1130, 1150; 2050, 2080; 3030, 3060; 4050, 4070; 5040, 5070; 6050, 6090 se vor înlocui cu instrucţiunile **IF** şi **GOTO** care simulează structura de iteraţie **CÎT TIMP** (v. conversaţia 5);
- Instrucţiunea de ştergere a ecranului din liniile: 1210, 2005, 3005, 4010, 5005, 6010 se va înlocui cu

```
2005 PRINT CHR$(147);
```

- Instrucţiunile **OPEN** de deschidere a fişierului de livrări în input din liniile 2040, 4035, 6030 se vor înlocui cu instrucţiunea

```
2040 OPEN, 2, 8, 2, "LIVRĂRI.DAT S, R"
```

- Instrucţiunea **OPEN** de deschidere a fişierului de livrări în output (pentru salvare) din linia 3010 se va înlocui cu instrucţiunea

```
OPEN 2, 8, 2, "LIVRĂRI.S, S, W"
```

- Instrucţiunea **CLOSE** de închidere a fişierului de livrări din liniile 2090, 3070, 4072, 6095 se va înlocui cu instrucţiunea

```
2090 CLOSE 2
3070 CLOSE 2
4072 CLOSE 2
6095 CLOSE 2
```

- Instrucţiunea **INKEY$** din linia 1140 se va înlocui cu instrucţiunea **GET**:

```
1135 GET A$
1140 OPŢIUNE$=A$
```

Formatul general al instrucţiunii este

Format general
GET [# ⟨nr. fişier⟩,] { ⟨variabilă numerică⟩ / ⟨variabilă şir⟩ }

unde, ⟨nr. fişier⟩ reprezintă numele fişierului; el este un întreg cuprins între 1 şi 255;

⟨variabilă numerică⟩ şi

⟨variabilă şir⟩ reprezintă valoarea (numerică/şir) transmisă de la tastatură.

Remarci

- Dacă ⟨variabilă numerică⟩ primeşte un caracter nenumeric se afişează eroare;
- Dacă ⟨variabila şir⟩ nu primeşte nici o valoare se consideră implicit, drept valoare şirul vid;
- Dacă ⟨variabila numerică⟩ nu primeşte nici o valoare se consideră implicit drept valoare zero.

- Instrucţiunea **INPUT #** din liniile 2060, 4052, 6070 se va înlocui cu instrucţiunea:

```
2060 INPUT #2, COD(I), CANT(I), PREŢ(I)
4052 INPUT #2, COD(C), CANT(C), PREŢ(C)
6070 INPUT #2, COD(I), CANT(I), PREŢ(I)
```

- Instrucţiunea **WRITE #** din linia 3040 se va înlocui cu instrucţiunea:

```
3040 PRINT #2, COD(C), CANT(C), PREŢ(C)
```

- Depistarea sfîrşitului unui fişier secvenţial se realizează prin instrucţiunile:

```
RS=ST
IF RS > 0 THEN CLOSE
```

unde, **ST** reprezintă starea fişierului (cuvînt cheie BASIC-COMMODORE).

Aplicaţie. Să se proiecteze şi realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea şi listarea unui fişier (secvenţial) privind nomenclatorul străzilor municipiului Bucureşti.

Informaţiile minimale necesare constituirii fondului de date sînt:

– cod stradă
– denumirea străzii
– sector
– oficiu poştal

Joc pe HC-85, TIM S, SPECTRUM

```
1 REM CHEIA
5 LET nt=0
10 DIM t(4,20)
20 LET a$="101001010111010101110"
30 LET flag=0
40 RANDOMIZE
50 FOR j=1 TO 3
60 LET r=INT(RND*19+1)
70 FOR k=1 TO 20
80 LET dis=k+r
90 IF dis<21 THEN GO TO 110
100 LET dis=dis-20
110 LET t(j, k)=VAL a$ (dis)
120 NEXT k
130 NEXT j
140 IF flag>0 THEN GO TO 180
150 FOR k=1 TO 20
160 LET t(4,k)=t(1,k)+t(2,k)+t(3,k)
170 NEXT k
180 LET flag=flag+1
190 IF flag>1 THEN GO TO 210
200 IF flag=1 THEN GO TO 50
210 CLS
220 LET tel=0
230 FOR k=1 TO 20
240 PRINT AT 10,k+2;t(1,k)
250 PRINT AT 11,k+2;t(2,k)
260 PRINT AT 12,k+2;t(3,k)
270 LET v=t(4,k)-t(3,k)-t(2,k)-t(1,k)
280 LET v=ABS(v)
290 IF v=0 THEN LET tel=tel+1
310 PRINT AT 14,k+2;v
320 NEXT k
330 IF tel=20 THEN GO TO 450
340 INPUT "LINIA=";j
350 INPUT "MUTAREA=";s
360 LET nt=nt+1
370 FOR i=1 TO s
380 LET h=t(j,1)
390 FOR k=1 TO 19
400 LET t(j,k)=t(j,k+1)
410 NEXT k
420 LET t(j,20)=h
430 NEXT i
440 GO TO 210
450 PRIN FLASH 1; "AŢI REUŞIT ÎN";
    nt; "ÎNCERCĂRI"
```

□ Particularităţi ale programării în limbajul BASIC-80

vol. 2, pag. 231

Cu mici diferenţe, programul conversaţiei este identic cu cel executat pe calculatorul AMSTRAD. În continuare vă invităm pe dvs. să le identificaţi ca apoi să le discutăm împreună.

După cum aţi putut constata, o primă diferenţă apare la instrucţiunea **OPEN** (v. liniile 2040, 3010, 4035, 6030). În limbajul BASIC-80, prin parametrul **⟨MOD⟩** (v. formatul general al instrucţiunii **OPEN**, conversaţia 7) se specifică modul de intrare/ieşire. Instrucţiunile 2040, 4035 şi 6030

```
2040 OPEN "I", #2, "LIVRĂRI.DAT"
```

deschid fişierul de livrări în input ("I"), iar instrucţiunea

```
3010 OPEN "O", #2, "LIVRĂRI.S"
```

deschide fişierul LIVRĂRI.S în output.

Instrucţiunea de ştergere a ecranului (v. liniile 1210, 2005, 3005, 4010, 5005, 6010) se va înlocui cu **CHR$** (24).

În BASIC-80 închiderea fişierelor se realizează cu **CLOSE** urmat de numărul fişierului.

```
2090 CLOSE #1
3070 CLOSE #2
4072 CLOSE #1
6095 CLOSE #1
```

Sperăm că aţi remarcat cum instrucţiunile **INPUT** din liniile 2060, 4052, 6080 s-au înlocuit cu:

```
2060 INPUT #1, COD(I), CANT(I), PREŢ(I)
4052 INPUT #1, COD(C), CANT(C), PREŢ(C)
6080 INPUT #1, COD(I), CANT(I), PREŢ(I)
```

În linia 3040 aţi găsit vreo deosebire faţă de programul implementat pe AMSTRAD?

```
3040 PRINT #2, COD(C), CANT(C), PREŢ(C)
```

În BASIC-80 pentru scrierea datelor într-un fişier secvenţial poate fi utilizată şi instrucţiunea **WRITE #**.

Formatul general al instrucţiunii este:

Format general
WRITE [#] ⟨număr-fişier⟩, ⟨listă-expresii⟩

unde, ⟨număr-fişier⟩ este numărul sub care fişierul a fost deschis în output ("O"); ⟨lista-expresii⟩ reprezintă şiruri sau expresii numerice, separate prin virgulă.

Observaţie. Diferenţa dintre **WRITE #** şi **PRINT #** constă în aceea că **WRITE #** inserează o virgulă după fiecare element introdus în fişier şi **[CR]** după ultimul element din listă.

O ultimă diferență apare la testarea sfîrșitului de fișier (secvențial). În BASIC-80 se utilizează tot funcția **EOF**, dar cu formatul

Format general
EOF (⟨număr-fișier⟩)

unde, ⟨număr-fișier⟩ reprezintă numărul fișierului precizat în instrucțiunile **OPEN, CLOSE, INPUT.**

Observație. **EOF** returnează valoarea –1 atunci cînd s-a atins sfîrșitul fișierului.

TEST

Încercați să treceți de la programul scris în limbajul BASIC-80 la programul scris în limbajul BASIC-AMSTRAD.

Aplicație. Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier (secvențial) privind cheltuielile unor secții de prelucrări (mecanice) dintr-o întreprindere constructoare de mașini.

Structura înregistrării fișierului de cheltuieli este:

– Cod secție
– Denumire secție
– Cheltuieli
  · ianuarie
  · februarie
  · martie
  · aprilie
  ·
  ·
  ·
  · decembrie

□ Crearea și întreținerea unui fișier de livrări în acces direct în limbajul BASIC-80

Limbajul BASIC-80 dispune și de facilități privind lucrul cu fișiere în acces direct. BASIC-80 pune la dispoziția utilizatorului trei instrucțiuni speciale: **FIELD #, LSET** și **RSET,** precum și un set de funcții de conversie.

În cele ce urmează vă prezentăm două programe pentru crearea (formatarea), respectiv întreținerea unui fișier de livrări în acces direct.

Creare fișier

Structura fișierului este: cod beneficiar, denumire beneficiar, cantitate livrată și preț.

```
10 REM "Formatare fişier LIV.DAT"
20 OPEN "R", #1, "LIV . DAT"
30 FIELD#1, 1 AS C1$, 20 AS C2$, 4 AS C3$, 7 AS C4$
40 LSET C1$=CHR$ (255)
50 FOR I=1 TO 98
60    PUT#1, I
70 NEXT I
80 CLOSE#1
90 END
```

Instrucţiunea FIELD

După deschiderea fişierului în acces direct LIV . DAT (v. linia 20), instrucţiunea **FIELD #** alocă un spaţiu, respectiv 1, 20, 4, 7 caractere variabilelor: C1$ (cod beneficiar), C2$ (denumire beneficiar), C3$ (cantitate livrată), C4$ (preţ).

Instrucţiunea **FIELD #** are următorul format:

Format general
FIELD [#] ⟨nr. fişier⟩, ⟨lungime cîmp⟩ AS ⟨var şir⟩ {,⟨lungime cîmp⟩ AS ⟨var. şir⟩ . . . }

în care: ⟨nr. fişier⟩ este numărul fixat pentru fişier la deschidere; ⟨lungime cîmp⟩ reprezintă numărul de caractere ale cîmpului.

Remarci

- Instrucţiunea **FIELD#** este o instrucţiune executabilă dar execuţia ei nu produce nici un transfer de informaţii, ci numai declară cîmpurile buffer-ului ca avînd nume de variabilă pentru identificarea lor ulterioară.
- Instrucţiunea **FIELD#** nu se foloseşte pentru variabilele utilizate în instrucţiunile **LET** sau **INPUT.**
- Numărul total al octeţilor alocaţi instrucţiunii **FIELD#** nu trebuie să depăşească lungimea înregistrării specificate la deschiderea fişierului în acces direct.

Instrucţiunea LSET

Instrucţiunea **LSET** din linia 40 realizează transferul efectiv al datei (CHR$(255)) în zona de alocare a variabilei.

Formatul instrucţiunii **LSET** este:

Format general
LSET ⟨var. şir⟩=⟨exp. şir⟩

unde: ⟨var. şir⟩ reprezintă orice variabilă simplă sau indexată de tip şir; ⟨exp. şir⟩ reprezintă orice expresie de tip şir a cărei valoare va face obiectul transferului (valorile numerice trebuie convertite în şiruri înainte de a fi transferate cu **LSET** – aliniază şirul la stînga şi **RSET** – aliniază şirul la dreapta).

Instrucţiunea PUT

În sfîrşit, instrucţiunea **PUT** (v. linia 60) scrie 98 de înregistrări (v. ciclul **FOR-NEXT** din liniile 50–70) în fişierul LIV . DAT.

Formatul instrucţiunii este:

Format general
PUT [#] ⟨nr. fişier⟩ [,⟨nr. înreg⟩]

unde: ⟨nr. înreg.⟩ reprezintă numărul înregistrării din fişier.

Observaţie. Dacă ⟨nr. înreg⟩ lipseşte, atunci înregistrarea va primi numărul următor celui dat ultimei înregistrări în fişier. ⟨nr. înreg⟩ este cuprins între 1 şi 32767.

Întreţinere fişier

Programul următor (cu greşeli!) actualizează fişierul LIV . DAT, prilej cu care vi se prezintă funcţia **MKS$** (converteşte un număr în simplă precizie într-un şir de 4 octeţi) din linia 515 şi instrucţiunea **GET #** (v. linia 100) care serveşte la transferarea unui articol din fişierul deschis, pe canalul indicat, în memoria internă, articolul fiind considerat ca o entitate. Funcţia **CVS** (v. linia 120) converteşte şirurile de cifre în valori numerice.

```
10 REM "Actualizare fişier LIV.DAT"
20 OPEN "R", #1, "LIV.DAT", 32
30 FIELD #1,1 AS C1$, 20 AS C2$, 4 AS
   C3$, 7 AS C4$
40 PRINT CHR$ (24)
50 PRINT "***ACTUALIZARE FIŞIER LIV.
   DAT***"
60 INPUT "Cod beneficiar"; CB
70 IF CB=0 THEN CLOSE#1 : END
80 IF CB>=1 AND CB<=98 AND CB=
   INT (CB) THEN 100
90 PRINT "Cod eronat" : GOTO 60
100 GET#1, CB
110 IF C1$=CHR$(255) THEN PRINT
   "Cod inexistent" : GOTO 300
120 DEN$=C2$ : CANT=CVS(C3$) :
   PREŢ=CVS(C4$)
130 PRINT "1. Denumire beneficiar";
   DEN$
140 PRINT "2 . Cantitatea livrată"; CANT
150 PRINT "3 . Preţ        "; PREŢ
160 PRINT INPUT "OK (Y/M/D)"; Y$
170 IF Y$="Y" THEN 40
180 IF Y$="M" THEN 200
190 IF Y$="D" THEN 400
195 GOTO 160
200 REM "Modificare"
210 INPUT "Număr linie="; I
220 IF I=0 THEN GOSUB 500 : GOTO 40
230 GOSUB 1000 : GOTO 210
300 REM "Creare"
310 INPUT "Creare un nou element
   (D/N) ";Y$
320 IF Y$="D" THEN 350
330 IF Y$="N" THEN 40
340 GOTO 310
350 FOR I=1 TO 3 : GOSUB 1000 : NEXT I
360 GOSUB 500 : GOTO 40
400 REM "Ştergere"
410 LSET C1$=CHR$(255) : PUT# 1, CB
420 PRINT "Înregistrare ştearsă" : INPUT
   "Tastează o cheie"; Y$
430 GOTO 40
500 REM "Scriere înregistrare"
510 LSET C1$=CHR$(0) : LSET C2$=
   DEN$
515 LSET C3$=MKS$(CANT) : LSET
   C4$=MKS$ (PREŢ)
520 PUT#1, CB
530 RETURN
1000 REM "Intrări subrutine"
1010 ON I GOSUB 1100, 1200, 1300 :
   RETURN
1100 INPUT "1. Denumire beneficiar";
   DEN$ : RETURN
1200 INPUT "2. Cantitatea livrată";
   CANT : RETURN
1300 INPUT "3. Preţ";
   PREŢ : RETURN
2000 END
```

Instrucţiunea GET

Formatul general al instrucţiunii este:

Format general
GET [#] ⟨nr. fişier⟩ [,⟨nr. înreg⟩]

unde: ⟨nr. fişier⟩ este numărul atribuit fişierului la deschidere;
⟨nr. înreg⟩ reprezintă numărul înregistrării de citit. Dacă este omis, se citeşte următoarea înregistrare din fişier (cel mai mare număr de înregistrări posibile este 32767).

☐ Particularităţi ale programării în limbajul BASIC-PLUS

Pentru scrierea programului BASIC–PLUS se vor avea în vedere următoarele **particularităţi**:

- limbajul BASIC-PLUS nu are implementată structura **WHILE**. Buclele **WHILE-WEND** (v. liniile 1020, 1230, 1130, 1150, 2050, 2080, 3030, 3060, 4050, 4070, 5040, 5070, 6050, 6090) din EXEMPLUL 8–PC (AMSTRAD) se vor simula cu **IF** şi **GO TO**;
- pentru deschiderea unui canal pentru un fişier deja existent se utilizează instrucţiunea **OPEN... FOR INPUT** (v. liniile 2040, 4035 şi 6030 din programul BASIC-AMSTRAD). Dacă fişierul indicat prin ⟨specificator fişier⟩ nu există pe suport, execuţia instrucţiunii va fi abandonată (se tipăreşte un mesaj de eroare);
- instrucţiunea **OPEN... FOR OUTPUT** presupune deschiderea unui canal pentru un nou fişier (v. linia 3010 din programul BASIC-AMSTRAD). Dacă fişierul indicat prin ⟨specificator fişier⟩ nu există acesta se creează pe suportul menţionat; dacă există, se creează un nou fişier;

Observaţie. Deschiderea canalului pentru prelucrarea unui fişier este automat însoţită de alocarea în memoria internă a spaţiului necesar bufferului de intrare/ieşire.

- instrucţiunea **CLOSE** provoacă închiderea unui canal de transfer deschis printr-o instrucţiune **OPEN** (v. liniile 2090, 3070, 4072, 6095 din programul BASIC-AMSTRAD);
- limbajul BASIC-PLUS nu admite instrucţiunea **INKEY$** (v. linia 1140 din programul BASIC-AMSTRAD); se recomandă utilizarea instrucţiunii **INPUT**;
- pentru transferul efectiv al datelor dintr-un fişier deschis pe un anumit canal de transfer în memoria internă se utilizează instrucţiunea **INPUT #** (v. liniile 2060, 4052, 6070 din programul BASIC-AMSTRAD). Fişierele care se exploatează prin **INPUT#** trebuie să aibă inserat caracterul "," între elementele unui articol.

Observaţie. Dacă ⟨expresie numerică⟩ este zero (v. formatul general al instrucţiunii **INPUT#**, conversaţia 7), suportul care se ia în considerare este terminalul utilizator.

Pentru citirea unui singur articol din fişierul asociat canalului, se utilizează instrucţiunea **INPUT LINE #** al cărei format general este:

Format general
INPUT LINE# ⟨expresie numerică⟩, ⟨variabilă şir⟩

unde, ⟨expresie numerică⟩ este o expresie numerică a cărei valoare (întreagă) cuprinsă între 0 şi 8 precizează numărul canalului de transfer; ⟨variabila şir⟩ este o variabilă de tip numeric căreia i se atribuie ca valoare textul articolului.

Reguli

- **INPUT LINE #** trebuie obligatoriu precedat de instrucţiunea **OPEN**;
- Caracterul "," işi pierde semnificaţia de separator al datelor articolului;
- Instrucţiunea **PRINT#** asigură transferul de date între memoria internă şi un suport orientat pe un articol (LP, TT etc.) sau un fişier (v. linia 3040 din programul BASIC-AMSTRAD);
- pentru prelucrarea fişierului ASCII secvenţiale se mai pot utiliza instrucţiunile: **PRINT#... USING** (instrucţiune de scriere a unui articol prin editarea sub un anumit format); **MAT INPUT#** (instrucţiune de citire a unui masiv dintr-un fişier);

MAT PRINT# (instrucțiune de scriere a unui masiv pe un fișier); **KILL** (instrucțiune de eliminare a unui fișier ASCII secvențial);

- Marcarea sfîrșitului fizic al unui fișier secvențial ASCII intră în sarcina utilizatorului. La exploatare se va testa sfîrșitul de fișier indicat prin program. În caz contrar se va semnala un mesaj de eroare (v. liniile 2050, 6050 din programul BASIC-AMSTRAD).

Aplicație. Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea, consultarea și listarea unui fișier (secvențial) de beneficiari ai unei întreprinderi constructoare de utilaj petrolier.

Informațiile minimale necesare constituirii fondului de date pentru beneficiarii întreprinderii sînt:
- Cod beneficiar
- Denumire beneficiar
- Adresă
- Cont bancă

□ Particularități ale programării în limbajul ABASIC

Pentru scrierea programului se vor avea în vedere următoarele **particularități**:

- instrucțiunea **OPEN,** de deschidere a fișierului este urmată (opțional) de caracterul "#" apoi de numărul fișierului și de numele fișierului ce respectă convențiile de sintaxă ARIEL (v. liniile 2040, 3010, 4035, 6030 din programul BASIC–AMSTRAD);
- în limbajul ABASIC nu există instrucțiunea **INKEY$;**
- pentru citirea datelor dintr-un fișier (secvențial) se utilizează instrucțiunile de citire: **INPUT # și LINPUT #;**
- pentru scrierea datelor într-un fișier (secvențial) se pot folosi instrucțiunile: **PRINT #, PRINT # . . . USING, WRITE #.** Instrucțiunile au același efect ca și cele de scriere directă la terminal;
- detectarea sfîrșitului de fișier se realizează folosind funcția

EOF (⟨expresie⟩)

care ia valoarea −1 atunci cînd s-a citit sfîrșitul fișierului identificat prin ⟨expresie⟩ sau 0 dacă sfîrșitul fișierului nu a fost detectat.

Aplicație. Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier (secvențial) privind nomenclatorul produselor unei întreprinderi de confecții.

Informațiile minimale necesare constituirii fondului de date sînt:
- cod produs
- denumire produs
- caracteristici
- preț
- anul fabricației.

TEMA 8

☐ **Răspundeţi prin DA sau NU la următoarele întrebări:**

– Instrucţiunea **INKEY$** interoghează tastatura.
– Dacă nu se acţionează nici o tastă, **INKEY$** returnează un şir vid.
– Instrucţiunea **ERASE** elimină variabilele tablou dintr-un program.
– În BASIC-AMSTRAD, instrucţiunea **INKEY** poate fi utilizată şi pentru numere întregi.
– Instrucţiunea **OPENIN** este specifică limbajului BASIC-PLUS.
– Funcţia **EOF** testează sfîrşitul de fişier.
– Instrucţiunea **CLOSEIN** închide fişierele BASIC-80.
– Cu instrucţiunea **ON ... GO TO** se realizează saltul la una din liniile de după **GO TO.**
– Instrucţiunea **INKEY$** este acceptată de limbajul BASIC-COMMODORE.
– **EOF** returnează valoarea –1 atunci cînd s-a depistat sfîrşitul unui fişier secvenţial.

☐ **Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:**

a) Funcţia **EOF** (⟨expresie⟩) poate fi utilizată în instrucţiunile ...., ..., ...

b) Instrucţiunea **INKEY$** este folosită pentru .... în timp ce instrucţiunea **INPUT** se utilizează pentru ....

c) Instrucţiunea **ERASE** este implementată în BASIC ...., ...

d) În BASIC-COMMODORE, depistarea sfîrşitului unui fişier secvenţial se realizează prin ...., ...

e) În limbajul BASIC-80 prin parametrul ⟨**MOD**⟩ al unei instrucţiuni **OPEN** se specifică ....

f) În limbajul BASIC-80 diferenţa dintre **WRITE#** şi **PRINT#** constă în ....

g) Instrucţiunea **OPEN ... FOR INPUT** din limbajul BASIC-PLUS presupune .... iar instrucţiunea **OPEN ... FOR OUTPUT** presupune ....

h) În limbajul .... instrucţiunea **INPUT LINE#** trebuie obligatoriu precedată de ....

i) Setul de instrucţiuni al limbajului BASIC-PLUS pentru prelucrarea fişierelor ASCII secvenţiale este următorul: ...., ..., ..., ..., ..., ..., ..., ..., ...

j) În secvenţa de instrucţiuni BASIC-PLUS:

```
10 OPEN "AA . A" FOR INPUT AS FILE 2 TO WRITE
20 OPEN "AA . B" FOR OUTPUT AS FILE 4 TO READ
30 PRINT# 3, M, N$
40 INPUT# 2, A, C%
```

instrucţiunile ...., ... sînt incorecte.

□ Scrieţi cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Să se gestioneze pe sistemele de calcul FELIX C, mini, micro şi calculatoare personale datele referitoare la personalul care locuieşte în municipiul Bucureşti. Informaţiile minimale necesare constituirii şi actualizării setului de date privind personalul care locuieşte în municipiul Bucureşti sînt:

- număr matricol
- nume
- prenume
- adresă
  - sector
  - stradă
  - număr
- data naşterii
  - zi
  - lună
  - an
- sex

Tranzacţiile care afectează conţinutul seturilor de date prezentate anterior sînt definite după cum urmează:

- tranzacţii de tip adăugări
- tranzacţii de tip modificări
- tranzacţii de tip ştergeri

FTRANZ

- tip tranzacţie
- număr matricol
- nume
- prenume
- adresă
  - sector
  - stradă
  - număr
- sex

tip tranzacţie avînd una din valorile:

13 = introducerea unei noi înregistrări;
14 = ştergerea unei înregistrări;
15 = modificarea conţinutului unei înregistrări.

Rapoartele de ieşire sînt următoarele:

1) **LISTA L1:** LISTA CU PERSONALUL CARE LOCUIEŞTE ÎN SECTORUL 1:

Conţinut:

- număr curent
- număr matricol
- nume şi prenume
- strada
- număr
- data naşterii
  - zi
  - luna
  - an
- sex

NR. LOCUITORILOR DIN SECTORUL 2: * * *
NR. LOCUITORILOR DIN SECTORUL 3: * * *
NR. LOCUITORILOR DIN SECTORUL 6: * * *

2) **LISTA L2**

Conţinut:

– nume – număr
– prenume – sector
– stradă – nume oraş

unde, nume-oraş este BUCUREŞTI.

Formatul listei L2 urmează să fie precizat de către fiecare dintre dumneavoastră.

b) Să se calculeze integrala definită

$$\int_{1.23}^{18.25} (x^2+x+1)dx$$

prin metoda trapezelor, dreptunghiurilor şi Simpson. Programul va fi ghidat de un meniu.

c) Să se determine prin metodele Newton-Raphson şi bisecţie soluţia ecuaţiei

$$x^3-2x-9=0$$

cunoscînd că o rădăcină a acestei ecuaţii este cuprinsă în intervalul (2,3÷2,5). Precizia este 10^{-5}. Programul va fi ghidat de un meniu.

d) Să se gestioneze pe mini, micro şi calculatoare personale datele referitoare la transformatoarele din posturile de transformare ce aparţin unui centru judeţean de distribuţie a energiei electrice. Informaţiile minimale necesare constituirii şi actualizării seturilor de date sînt: cod post transformare; caracteristici transformator; anul punerii în funcţiune a transformatorului; seria; puterea; grupa de conexiuni; tensiunea de scurtcircuit; bateriile de condensatoare pentru compensarea factorului de putere; circuitele de iluminat public; circuitele de forţă; adresa.

Programul BASIC va fi ghidat de un meniu care să permită realizarea următoarelor funcţiuni: **1.** help; **2.** creare fişier; **3.** adăugare înregistrări; **4.** modificare cîmpuri; **5.** ştergere înregistrări; **6.** modificare structură fişier; **7.** listare fişier; **8.** end.

Observaţie. Programul este operaţional la I.R.E Ploieşti.

e) Există în practica multor I.R.E.-uri din ţară de a raporta situaţiile referitoare la consumul de energie la nivelul judeţului. Raportarea înglobează situaţia consumurilor de energie pe ministere, situaţia energiei suplimentare generate de autoproducători cît şi situaţia consumului de energie al principalilor consumatori la nivelul judeţului.

Să se gestioneze pe mini, micro şi calculatoare personale datele referitoare la consumul de energie electrică la nivel de judeţ. Informaţiile minimale necesare constituirii şi actualizării seturilor de date sînt: cod mi-

nister; denumire minister, energie totală contur judeţ; puterea totală pe palierul 2 (3, 4) pe judeţ; numărul maxim de ministere pe judeţ; numărul maxim de întreprinderi ale ministerului; energia totală pe paliere pe întreprindere; energia pe palierul 2 (3, 4) pe întreprindere. Programul BASIC va fi ghidat de un meniu şi va genera raportul zilnic ce conţine: energia totală contur judeţ; puterile totale pe palierul 1 (2, 3, 4) pe judeţ; energia totală pe judeţ; puterile totale pe palierul 1 (2, 3, 4) pe judeţ.

Observaţie. Programul este operaţional la I.R.E. Ploieşti.

☐ **Se reia problema din conversaţia precedentă,** complicînd-o după cum urmează.

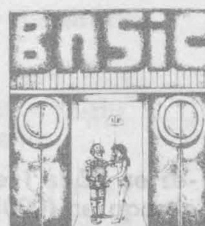
Să se proiecteze şi să se realizeze un program BASIC ghidat de un meniu care să permită realizarea următoarelor funcţiuni: **I.** Încarcă fişier JUCĂRII.DAT; **S.** Salvează fişier; **C.** Căutare în fişier; **L.** Listare fişier; **F.** Stop.

☐ **Se reia problema din conversaţia precedentă,** complicînd-o astfel. Să se proiecteze şi să se realizeze un program BASIC ghidat de un meniu cu următoarea structură:

1. Încărcare fişier CONSUMURI . DAT; **2.** Actualizare fişier; **3.** Listare fişier; **4.** Stop.

Indicaţie. Se vor utiliza subrutinele din EXEMPLELE 8 – PC, m, M, F.

Proiectarea și realizarea unui program BASIC-PLUS ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier masiv virtual de livrări. Instrucțiunile DIM, OPEN... AS VIRTUAL și CLOSE. JOCURI, APLICAȚII și TESTE pentru cititor



□ De la problemă la program

Vom relua problema din conversaţiile 7 şi 8, cu deosebirea că fişierul de livrări va fi organizat ca un *fişier masiv virtual.* Fişierele masive virtuale sînt masive rezidente pe un disc magnetic, ca extensie a memoriei interne, ale căror dimensiuni sînt limitate numai de mărimea spaţiului liber pe suportul fizic (disc). Faţă de masivele reale (declarate în memoria calculatorului) nu există nici o restricţie privind utilizarea acestora.

Limbajul BASIC-PLUS asigură accesul la oricare din elementele fişierului, indiferent de locul pe care acestea le ocupă în fişierul de pe disc.

În cadrul acestei conversaţii vom proiecta şi realiza un program (BASIC-PLUS) ghidat de un meniu pentru: crearea fişierului masiv virtual de livrări, adăugarea de beneficiari noi (funcţiunea nu apare în conversaţiile 7 şi 8), căutarea în fişier, actualizare fişier, listare fişier, sfîrşit execuţie program.

Funcţiunea de „adăugare beneficiari noi" realizează, după cum precizează şi numele, extinderea fişierului cu beneficiari al căror cod nu există

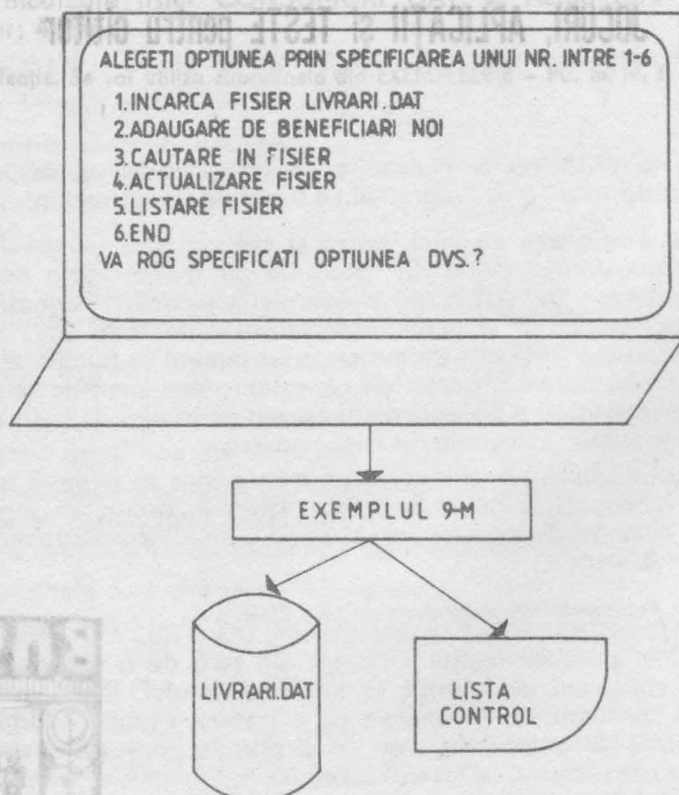


Fig. 9.1. Schema de sistem.

în fișier. Prin specificarea unui număr cuprins între 1 și 6 utilizatorul poate opta pentru una din funcțiunile precizate.

Deoarece programul are ca scop ilustrarea facilităților limbajului BASIC-PLUS pentru prelucrarea fișierelor masive virtuale de tip întreg, real și datorită faptului că sînteți experimentați cu conceptele generale de analiză și proiectare, vă vom prezenta numai schema de sistem a aplicației (v. figura 9.1).

Astfel, veți putea să vă concentrați mai mult asupra instrucțiunilor: **DIM#, OPEN . . . AS VIRTUAL.**

☐ Codificarea în limbajul BASIC-PLUS

vol. 2, pag. 234

Meniul conversației

Programul este alcătuit dintr-un program principal (liniile 100–130) care apelează o subrutină (140) de prezentare a meniului.

```
100 REM **ÎNTREȚINERE FIȘIER LIVRĂRI. DAT**
110 DIM # 1,C(100), Q(100), P(100)
120 GOSUB 140
130 END

140 REM SUBRUTINA DE PREZENTARE A MENIULUI
150 PRINT
160 PRINT "ALEGEȚI OPȚIUNEA PRIN SPECIFICAREA UNUI NR. ÎNTRE 1–6"
170 PRINT
180 PRINT " 1. ÎNCARCĂ FIȘIER LIVRĂRI. DAT"
190 PRINT " 2. ADĂUGARE DE BENEFICIARI NOI"
200 PRINT " 3. CĂUTARE ÎN FIȘIER"
210 PRINT " 4. ACTUALIZARE FIȘIER"
220 PRINT " 5. LISTARE FIȘIER"
230 PRINT " 6. END"
240 INPUT "VĂ ROG SPECIFICAȚI OPȚIUNEA DVS.", O%
250 IF O%<7 THEN 270
260 PRINT "***OPȚIUNE ERONATĂ***" : GO TO 150
270 IF O%=1 THEN GOSUB 350
280 IF O%=2 THEN GOSUB 510
290 IF O%=3 THEN GOSUB 680
300 IF O%=4 THEN GOSUB 820
310 IF O%=5 THEN GOSUB 1000
320 GO TO 340
330
340 RETURN
350 REM*1. ÎNCARCĂ FIȘIER LIVRĂRI. DAT
360 OPEN "LIVRĂRI. DAT" AS VIRTUAL FILE 1
370 FOR K=1 TO 100
380     C(K)=0 : Q(K)=0 : P(K)=0
390 NEXT K
```

În funcție de opțiunea indicată se tastează un număr (între 1 și 5), se selectează una din cele cinci subrutine (350, 510, 680, 820, 1000) ce aparțin subrutinei de prezentare a meniului (140). Pentru oprirea execuției programului se tastează cifra 6.

Fişiere masive virtuale. Structură, acces

Un fişier masiv virtual poate fi imaginat ca o succesiune de zone. Un masiv virtual apare utilizatorului ca o succesiune de blocuri cu lungimea de 512 octeţi fiecare. Un bloc conţine un număr întreg de elemente ale masivului (v. tabelul 9.1).

Tabelul 9.1

Tip masiv	Lungime element (octeţi)	Număr elemente/bloc
Întreg	2	256
Real	4	128
Şir	n	512/n

Reguli

- Lungimea (m) unui element de masiv şir este obligatoriu o putere a lui 2, cuprinsă între 2 şi 512;
- Referirea unui element de masiv se realizează în cadrul programului prin expresii de indici care se transformă, prin mecanismul de tratare a masivelor virtuale, într-o pereche de adrese relative;
- Bufferul alocat de calculator pentru un fişier masiv virtual este de dimensiunea unui bloc disc (512 octeţi);
- În fişierul masiv virtual nu se păstrează nici o informaţie privind numele sau dimensiunile masivelor existente.

Declararea masivului virtual prin instrucţiunea DIM#

110 **DIM**#1,C(100), Q(100), P(100)

Programul prelucrează masivele virtuale C, Q şi P. Pentru prelucrarea acestora este necesară declararea mai întîi a celor trei masive virtuale prin instrucţiunea **DIM**#. În cadrul instrucţiunii se precizează canalul de transfer (prin care se realizează transferul elementelor între memoria internă şi fişierul disc suport) şi lista masivelor.

Formatul general al instrucţiunii este:

Format general
DIM # ⟨expresie numerică⟩ , ⟨listă⟩

în care:

⟨expresie numerică⟩ este orice expresie numerică cu valoarea întreagă cuprinsă între 1 şi 8 ce indică numărul canalului de transfer;

⟨listă⟩ este o listă de elemente indicînd numele masivelor şi dimensiunile acestora.

Reguli

- Un fişier disc poate fi suportul mai multor masive virtuale de acelaşi tip sau de tipuri diferite (în linia 110 cele trei masive: C, Q şi P au acelaşi fişier disc, deschis pe canalul 1).

- Şirurile-elemente ale unui masiv virtual de tip şir (nenumeric) au aceeaşi lungime fixă, declarată prin **DIM#** (v. conversaţia 10).
- Dacă **DIM#** nu conţine declaraţia de lungime, aceasta se consideră implicit egală cu 16.

Opţiunea 1. Încarcă fişier LIVRĂRI . DAT

Subrutina realizează funcţiunea de creare a fişierului masiv virtual LIVRĂRI . DAT. Pentru început se deschide fişierul suport al masivului virtual pe canalul de transfer 1, prin instrucţiunea **OPEN.**

Instrucţiunea OPEN . . . AS VIRTUAL

Pentru a obţine accesul la fişierul disc, suport al masivelor virtuale C, Q, P programul trebuie să conţină o instrucţiune **OPEN** de deschidere a canalului de transfer (1), asociat fişierului „LIVRĂRI . DAT".

```
360 OPEN "LIVRĂRI.DAT" AS VIRTUAL FILE 1
```

Formatul general al instrucţiunii **OPEN . . . AS VIRTUAL** este:

Format general
OPEN ⟨specificator fişier⟩ **[FOR INPUT | OUTPUT AS VIRTUAL FILE** ⟨nr. canal⟩ **[TO MODIFY] [,FILESIZE** ⟨lungime fişier⟩]

în care parametrii ⟨specificator fişier⟩, ⟨nr. canal⟩, ⟨lungime fişier⟩ şi opţiunile **INPUT | OUTPUT** au aceeaşi semnificaţie ca la instrucţiunea **OPEN** (v. conversaţia 7).

Reguli

- Pentru un fişier masiv virtual, modul de deschidere acceptat este **MODIFY.** Modul **MODIFY** asigură accesul pentru citire, scriere, modificare.
- Pentru scrierea de noi articole, fişierul poate fi extins, pas cu pas, prin alocări succesive de spaţiu suplimentar pe disc.
- Extensia fişierului pas cu pas poate fi evitată atribuind o anumită valoare (de regulă zero) ultimului element din ultimul masiv declarat pe fişierul respectiv (metoda se aplică numai atunci cînd se cunoaşte lungimea finală a fişierului — v. şi opţiunea **FILESIZE**).
- Masivul virtual (numeric, şir) nu este niciodată iniţializat (cu zero, respectiv şir vid).

Iniţializarea masivului virtual

Conform ultimei reguli listate (v. instrucţiunea **OPEN . . . AS VIRTUAL**) intră în sarcina noastră să iniţializăm (cu zero), cele trei masive numerice: C, Q şi P.

```
370 FOR K=1 TO 100
380   C(K)=0 : Q(K)=0 : P(K)=0
390 NEXT K
```

TEST

Scrieți secvența de instrucțiuni de mai sus utilizînd instrucțiunea matricială **MAT... ...ZER.**

Observație. Conform ultimei reguli de la instrucțiunea **OPEN... AS VIRTUAL,** cele trei masive au fost extinse prin atribuirea lui zero ultimului element al masivelor.

De la tastatură se introduc: COD, CANTITATE, PREȚ pentru cel mult 99 de beneficiari (v. liniile 400–500).

```
400 INPUT "COD : ";K%
410 IF K%>99 GO TO 400
420 IF K%=99 GO TO 470
430 INPUT "CANTITATE : "; C2
440 INPUT "PREȚ : "; C3
450 C(K%)=K% : Q(K%)=C2 : P(K%)=C3
460 GO TO 400
470 CLOSE 1
480 PRINT
490 PRINT "===FIȘIER ÎNCĂRCAT==="
500 RETURN
```

Opțiunea 2. Adăugare de beneficiari noi

Subrutina realizează introducerea în fișier a noi beneficiari. Subrutina testează codul beneficiarului. În caz de eroare – codul este mai mare sau egal cu 99 se afișează mesajul "COD BENEFICIAR ERONAT". În situația cînd se introduce un cod deja existent în fișier se va tipări "BENEFICIAR EXISTENT – OPERAȚIE NEPERMISĂ".

```
510 REM*2. ADĂUGĂRI DE NOI ÎNREGISTRĂRI
520 OPEN "LIVRĂRI.DAT" AS VIRTUAL FILE 1
530 INPUT "PRECIZAȚI CODUL BENEFICIARULUI : "; K%
540 IF K%<99 GO TO 570
550 PRINT "COD BENEFICIAR ERONAT"
560 GO TO 530
570 IF C(K%)=0 THEN 600
580 PRINT "BENEFICIAR EXISTENT – OPERAȚIE NEPERMISĂ"
590 GO TO 650
600 INPUT "CANTITATE : "; C2
610 INPUT "PREȚ : "; C3
620 C(K%)=K% : Q(K%)=C2 : P(K%)=C3
630
640
650 CLOSE 1
660 PRINT "===TERMINAT ADĂUGARE==="
670 RETURN
```

Instrucțiunea CLOSE

```
650 CLOSE 1
```

Instrucțiunea (**CLOSE**) comandă închiderea canalului de transfer (1). De notat că această închidere este obligatorie.

Formatul general al instrucţiunii este:

Format general
CLOSE ⟨nr. canal⟩

unde, ⟨nr. canal⟩ este numărul canalului de transfer.

Regulă. Eliberarea canalelor de transfer prin care au fost prelucrate masivele virtuale poate fi realizată şi cu instrucţiunile **END/CHAIN** cu observaţia că, în acest caz, pot fi pierdute ultimele modificări operate în masivele respective.

Opţiunea 3. Căutare în fişier

Subrutina realizează funcţia de regăsire a unui beneficiar după cod. În urma căutării în fişier, subrutina afişează un mesaj corespunzător găsirii sau nu a beneficiarului indicat.

```
680 REM * 3. CĂUTARE ÎN FIŞIER
690 OPEN "LIVRĂRI.DAT" AS VIRTUAL FILE 1
700 INPUT "PRECIZAŢI CODUL BENEFICIARULUI : "; K%
710 IF K%<99 GO TO 740
720 PRINT "COD BENEFICIAR ERONAT"
730 GO TO 700
740 A$="NU A"
750 IF C(K%)=0 THEN 770
760 A$="A"
770 CLOSE 1
780 REM BENEFICIARUL A FOST SAU NU GĂSIT ÎN FIŞIER
790 PRINT
800 PRINT "BENEFICIARUL CU CODUL"; K%; A$; "FOST GĂSIT ÎN FIŞIER"
810 RETURN
```

Opţiunea 4. Actualizare fişier

Subrutina actualizează cele două cîmpuri CANTITATE şi PREŢ corespunzătoare codului beneficiarului indicat. Atunci cînd beneficiarul nu se află în fişier se afişează mesajul "BENEFICIARUL * * * NU ESTE PREZENT ÎN FIŞIER".

```
820 REM*4. ACTUALIZARE FIŞIER
830 INPUT "PENTRU CE BENEFICIAR DORIŢI ACTUALIZAREA"; K%
840 IF K%<99 GO TO 870
850 PRINT "COD BENEFICIAR ERONAT"
860 GO TO 830
870 OPEN "LIVRĂRI DAT." AS VIRTUAL FILE 1
880 IF C(K%)=0 THEN 970
890 G%=1
900 PRINT "LA BENEF. : "; K%; "ESTE CANT. : "; Q(K%);" ŞI PREŢUL : "; P(K%)
910 PRINT "SPECIFICAŢI NOILE VALORI:"
920 INPUT "CANTITATE"; C2
930 INPUT "PREŢ"; C3
940 Q(K%)=C2 : P(K%)=C3
950
960 IF G%=1 THEN 980
970 PRINT "BENEFICIARUL"; K%; "NU ESTE PREZENT ÎN FIŞIER"
980 CLOSE 1
990 RETURN
```

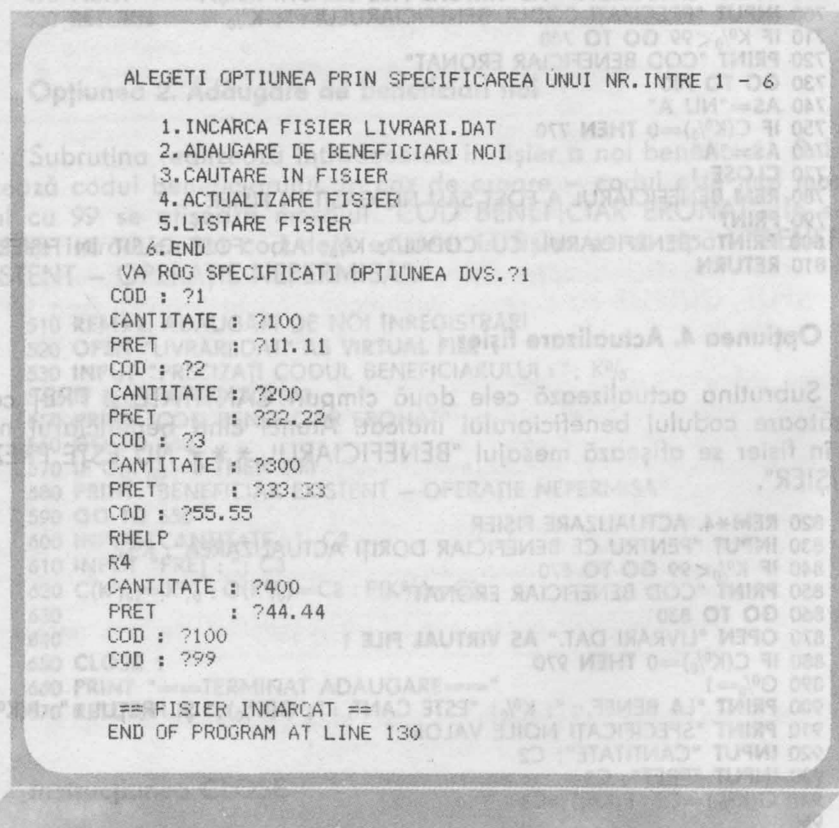
Opţiunea 5. Listare fişier

Subrutina tipăreşte pe ecranul monitorului întregul conţinut al fişierului.

```
1000 REM * 5. LISTARE FIŞIER
1010 OPEN "LIVRĂRI.DAT" AS
     VIRTUAL FILE 1
1020 PRINT "COD BENEF."; TAB (30);
     "CANTITATE"; TAB (50); "PREŢ"
1030 FOR I=1 TO 100
1040    IF C(I)=0 THEN 1060
1050    PRINT C(I); TAB (30); Q(I);
        TAB (50); P(I)
1060 NEXT I
1070 CLOSE 1
1080 PRINT : PRINT "LISTARE
     TERMINATĂ"
1090 RETURN
```

Rezultatele execuţiei programului

Încărcarea fişierului LIVRĂRI . DAT, adăugarea de beneficiari noi, căutarea în fişier, actualizarea fişierului, listarea fişierului, oprirea execuţiei programului sînt ilustrate în figura 9.2.



```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

        1.INCARCA FISIER LIVRARI.DAT
        2.ADAUGARE DE BENEFICIARI NOI
        3.CAUTARE IN FISIER
        4.ACTUALIZARE FISIER
        5.LISTARE FISIER
        6.END
   VA ROG SPECIFICATI OPTIUNEA DVS.?1
   COD : ?1
   CANTITATE : ?100
   PRET      : ?11.11
   COD : ?2
   CANTITATE : ?200
   PRET      : ?22.22
   COD : ?3
   CANTITATE : ?300
   PRET      : ?33.33
   COD : ?55.55
   RHELP
   R4
   CANTITATE : ?400
   PRET      : ?44.44
   COD : ?100
   COD : ?99

   == FISIER INCARCAT ==
   END OF PROGRAM AT LINE 130
```

a)

Fig. 9.2. Opţiuni: *a)* opţiunea 1 (la introducerea codului 55.55 de la terminal, s-a editat caracterul „R" deoarece s-a introdus pentru o variabilă întreagă (K%) o valoare reală; s-a tastat în continuare 4);

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6
   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?5
COD BENEF.                 CANTITATE           PRET
 1                          100               11.11
 2                          200               22.22
 3                          300               33.33
 4                          400               44.44

LISTARE TERMINATA
END OF PROGRAM AT LINE 130
:
:
:
```

b)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6
   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?2
PRECIZATI CODUL BENEFICIARULUI : ?8
CANTITATE : ?800
PRET      : ?88.88
 == TERMINAT ADAUGARE ==

END OF PROGRAM AT LINE 130
```

c)

Fig. 9.2. *b)* opţiunea 5; *c)* opţiunea 2;

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?5
COD BENEF.                   CANTITATE          PRET
 1                             100              11.11
 2                             200              22.22
 3                             300              33.33
 4                             400              44.44
 8                             800              88.88

LISTARE TERMINATA
END OF PROGRAM AT LINE 130
```

d)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?2
PRECIZATI CODUL BENEFICIARULUI : ?7
   CANTITATE : ?700
   PRET      : ?77.77
    == TERMINAT ADAUGARE ==
   END OF PROGRAM AT LINE 130
```

e)

Fig. 9.2. *d)* opţiunea 5; *e)* opţiunea 2;

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

    1.INCARCA FISIER LIVRARI.DAT
    2.ADAUGARE DE BENEFICIARI NOI
    3.CAUTARE IN FISIER
    4.ACTUALIZARE FISIER
    5.LISTARE FISIER
    6.END
 VA ROG SPECIFICATI OPTIUNEA DVS.?5
COD BENEF.                 CANTITATE            PRET
 1                            100              11.11
 2                            200              22.22
 3                            300              33.33
 4                            400              44.44
 7                            700              77.77
 8                            800              88.88

LISTARE TERMINATA
END OF PROGRAM AT LINE 130
```

f)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

    1.INCARCA FISIER LIVRARI.DAT
    2.ADAUGARE DE BENEFICIARI NOI
    3.CAUTARE IN FISIER
    4.ACTUALIZARE FISIER
    5.LISTARE FISIER
    6.END
 VA ROG SPECIFICATI OPTIUNEA DVS.?2
PRECIZATI CODUL BENEFICIARULUI : ?100
COD BENEFICIAR ERONAT
PRECIZATI CODUL BENEFICIARULUI : ?99
COD BENEFICIAR ERONAT
PRECIZATI CODUL BENEFICIARULUI : ?98
CANTITATE : ?9800
PRET      : ?98.98
 == TERMINAT ADAUGARE ==
END OF PROGRAM AT LINE 130
```

g)

Fig. 9.2. *f)* opţiunea 5; g) opţiunea 2;

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
 VA ROG SPECIFICATI OPTIUNEA DVS.?5
COD BENEF.                  CANTITATE          PRET
1                           100               11.11
2                           200               22.22
3                           300               33.33
4                           400               44.44
7                           700               77.77
8                           800               88.88
98                          9800              98.98

LISTARE TERMINATA
END OF PROGRAM AT LINE 130
```

h)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
 VA ROG SPECIFICATI OPTIUNEA DVS.?3
PRECIZATI CODUL BENEFICIARULUI : ?98

 BENEFICIARUL CU CODUL  98 A FOST GASIT IN FISIER
END OF PROGRAM AT LINE 130
```

i)

Fig. 9.2. *h)* opțiunea 5; *i)* opțiunea 3;

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
 VA ROG SPECIFICATI OPTIUNEA DVS.?3
PRECIZATI CODUL BENEFICIARULUI : ?15

 BENEFICIARUL CU CODUL 15 NU A FOST GASIT IN FISIER
END OF PROGRAM AT LINE 130
```

j)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?4
PENTRU CE BENEFICIAR DORITI ACTUALIZAREA  ?15
BENEFICIARUL 15 NU ESTE PREZENT IN FISIER
END OF PROGRAM AT LINE 130
```

k)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
 VA ROG SPECIFICATI OPTIUNEA DVS.?4
 PENTRU CE BENEFICIAR DORITI ACTUALIZAREA ?4
LA BENEF.: 4 ESTE CANT.: 400 SI PRETUL : 44.44
SPECIFICATI NOILE VALORI :
                       CANTITATE?400.44
                       PRET      ?44.444
END OF PROGRAM AT LINE 130
```

l)

Fig. 9.2. *j)* opţiunea 3; *k)* opţiunea 4; *l)* opţiunea 4;

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?5
COD BENEF.              CANTITATE            PRET
1                       100                  11.11
2                       200                  22.22
3                       300                  33.33
4                       400.44               44.444
7                       700                  77.77
8                       800                  88.88
98                      9800                 98.98

LISTARE TERMINATA
END OF PROGRAM AT LINE 130
```

m)

```
ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

   1.INCARCA FISIER LIVRARI.DAT
   2.ADAUGARE DE BENEFICIARI NOI
   3.CAUTARE IN FISIER
   4.ACTUALIZARE FISIER
   5.LISTARE FISIER
   6.END
VA ROG SPECIFICATI OPTIUNEA DVS.?6
END OF PROGRAM AT LINE 130
```

n)

Fig. 9.2. *m)* opţiunea 5; *n)* opţiunea 6.

Aplicaţie. Să se realizeze specificaţiile de programare, documentaţia de proiectare şi programul BASIC–PLUS pentru crearea şi întreţinerea unui fişier masiv virtual de strunguri. Structura fişierului este următoarea: cod, uzină constructoare, puterea, diametrul maxim al piesei prelucrate şi lungimea dintre vîrfuri. Exemplu: 100, IMU ARAD, SN 400 7.5, 400, 1000; 200, IMU ARAD, SNA 320, 5, 320, 750.

Joc pe HC-85, TIM S, SPECTRUM

```
10 REM CUTIA NEAGRĂ
20 DIM b (10, 10, 10)
30 GO TO 1000
100 REM INTRARE TIR
110 PRINT "INTRARE TIR"
120 GO SUB 500
130 LET dx=(px=2)-(px=9)
140 LET dy=(py=2)-(py=9)
150 LET dz=(pz=2)-(pz=9)
160 IF (dx=0) AND (dy=0) AND (dz=0)
    THEN GO TO 120
170 RETURN
200 REM CALCUL REZULTAT
210 FOR x=-1 TO 1
220 FOR y=-1 TO 1
230 FOR z=-1 TO 1
240 IF b(px+x, py+y, pz+z) < > 1
    THEN GO TO 280
250 LET dx=dx-x
260 LET dy=dy-y
270 LET dz=dz-z
280 NEXT z
290 NEXT y
300 NEXT x
310 LET dx=SGN dx: LET dy=SGN dy:
    LET dz=SGN dz
320 LET px=px+dx: LET py=py+dy:
    LET pz=pz+dz
330 IF (px=1)+(px=10)+(py=1)+
    (py=10)+(pz=1)+(pz=10)=0
    THEN GO TO 210
340 PRINT "REZULTAT:"; px-dx-1;
    " "; py-dy-1; " "; pz-dz-1
350 RETURN
400 REM INTRARE ÎNCERCARE
410 PRINT "INTRARE POZIŢIE ATOM"
420 GO SUB 500
430 IF b(px, py, pz)=1 THEN PRINT
    "BRAVO!" : RETURN
440 PRINT "FALS!"
450 RETURN
500 REM INTRARE COORDONATE
510 PRINT "TASTAŢI COORDONATELE"
520 INPUT "P1="; px: LET px=px+1
530 IF px<2 OR px>9
    THEN GO TO 520
540 INPUT "P2="; py : LET py=py+1
550 IF py<2 OR py>9 THEN GO TO 540
560 INPUT "P3="; pz: LET pz=pz+1
570 IF pz<2 OR pz>9 THEN GO TO 560
580 PRINT "INTRARE:"; px-1;" ";
    py-1;" "; pz-1
590 RETURN
1000 REM PROGRAM PRINCIPAL
1010 FOR a=1 TO 5
1020 LET px=INT (RND * 6+3) : LET
     py=INT (RND * 6+3) : LET pz=
     INT (RND * 6+3)
1025 IF b(px, py, pz)=1 THEN
     GO TO 1020
1030 LET b(px, py, pz)=1
1035 NEXT a
1040 GO SUB 100: REM INTRARE TIR
1050 GO SUB 200: REM CALCUL
     REZULTAT
1060 INPUT "TIR SAU POZIŢIE ATOM
     ? (t/p) "; t$
1070 IF t$<>"t" AND
     t$< >"p" THEN GO TO 1060
1080 IF t$="t" THEN GO TO 1040
1090 GO SUB 400: REM INTRARE
     ÎNCERCARE
1100 GO TO 1060
```

TEMA 9

□ Răspundeţi prin DA sau NU la următoarele întrebări:

– În BASIC-PLUS se pot defini masive virtuale rezidente pe un fişier disc, ca extensie a memoriei interne.

– Masivele virtuale pot fi utilizate fără nici o restricţie, comparativ cu masivele reale.

– Prin utilizarea masivelor virtuale se extinde memoria internă.

– Mărimea bufferului din memoria internă, în cazul unui fişier masiv virtual este diferită de dimensiunea unui bloc disc.

– În fişierul masiv virtual nu este păstrată nici o informaţie privind numele masivului conţinut.

– În fişierul masiv virtual se păstrează informaţii privind dimensiunile masivelor conţinute.

– Numărul de elemente/bloc pentru un masiv de tip întreg (%) este 256.

– Instrucţiunea **OPEN** ... **AS VIRTUAL** deschide canalul de transfer asociat unui fişier masiv virtual.

– Masivele virtuale numerice sînt automat iniţializate cu zero.

– Masivele virtuale şir nu sînt niciodată iniţializate.

– Cifra 1 din instrucţiunea

```
110 DIM # 1, C2(100), C(100), P(100)
```

indică numărul canalului de transfer.

☐ **Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:**

a) Masivele reale se declară în .... iar masivele virtuale ....

b) Lungimea unui element de masiv şir este o putere a lui ....

c) Pentru prelucrarea, prin program, a unui masiv virtual, trebuie realizate următoarele funcţiuni: ...., ..., ...

d) Declararea masivului virtual se realizează prin instrucţiunea........ .

e) Deschiderea fişierului suport al masivului virtual pe un canal de transfer se realizează prin instrucţiunea ....

f) Închiderea canalului de transfer cu instrucţiunea .... într-un program BASIC-PLUS este ....

g) Pentru un fişier masiv virtual, singurul mod de deschidere acceptat este modul ....

h) Cînd se doreşte scrierea unor noi articole în fişier, fără a cunoaşte lungimea finală a fişierului, se poate extinde fişierul, pas cu pas, prin .... iar în cazul că se ştie lungimea finală a fişierului se ....

i) Masivele virtuale numerice trebuie iniţializate prin program cu .... iar masivele virtuale şir trebuie iniţializate cu ....

j) În secvenţa de instrucţiuni:

```
10 DIM #1, A(100)
20 OPEN "A" AS VIRTUAL FILE 2
30 CLOSE 1
```

s-au strecurat următoarele greşeli: ...., ..., ...

k) În cadrul următorului program de creare a unui fişier masiv virtual de articole (cel mult 29) structurat în: COD, CANTITATE

```
10 DIM #1, C1(30), C(30)
20 OPEN "A.DAT" AS VIRTUAL FILE 1
```

```
30 C(30)=0
40 INPUT "COD : "; K%
50 IF K% > 29 GO TO . .
60 IF K%=29 GO TO . . .
70 INPUT "CANTITATE : " ; C2
80 C1(K)=K% : C(K%)=C2
90 GO TO 40
100 CLOSE 2
110 PRINT
120 PRINT "FIŞIERUL A . DAT A FOST CREAT"
130 END
```

s-au strecurat greşeli în liniile .... , ... , ...

În linia 50 instrucţiunea **GOTO** este urmată de eticheta .... iar în linia 60 trebuie adăugat după instrucţiunea **GO TO** numărul de linie ....

Instrucţiunea de atribuire din linia 30:

```
30 C(30)=0
```

a fost scrisă pentru ....

În acest caz s-a aplicat regula: cînd se cunoaşte lungimea finală a fişierului, se poate evita extensia pas cu pas printr-o singură extensie a fişierului atribuind (de regulă) valoarea .... , ... element din ultimul masiv declarat în fişierul respectiv.

☐ Scrieţi cîte un program BASIC-PLUS pentru fiecare din problemele de mai jos:

a) Să se creeze un fişier masiv virtual de livrări avînd structura: COD BENEFICIAR, CANTITATE, PREŢ.

b) Să se creeze şi să se întreţină un fişier masiv virtual ce conţine zilele onomastice ale celor dragi. Se va realiza un program BASIC-PLUS ghidat de un meniu.

c) Să se creeze şi să se întreţină un fişier masiv virtual ce conţine rezultatele meciurilor de rugbi susţinute de echipele divizionare A. Se va realiza un program ghidat de un meniu.

☐ **Să se gestioneze** şi să se întreţină pe minicalculatoarele INDEPENDENT şi CORAL fondul de date privind cantităţile lunare de jucării fabricate de Întreprinderea URSULEŢUL.

Se va proiecta şi realiza un program BASIC-PLUS ghidat de un meniu pentru crearea, actualizarea şi listarea fişierului masiv virtual JUCĂRII . DAT. Structura înregistrării este cea cunoscută (v. conversaţiile precedente).

Observaţie. Se vor utiliza subrutinele din EXEMPLUL 9-M.

☐ **Să se proiecteze şi să se realizeze** un program BASIC-PLUS ghidat de un meniu pentru crearea şi întreţinerea fişierului masiv virtual CONSUMURI . DAT a cărui structură a fost definită în conversaţiile precedente.

SOLUŢIA TEMEI 9

☐ **Răspunsurile sînt:** DA, DA, DA, NU, DA, NU, DA, DA, NU, DA, DA.

☐ **Cuvintele lipsă sînt:**

a) memoria internă/pe fişiere disc; b) 2, cuprinsă între 2 şi 512; c) declararea masivului virtual, deschiderea fişierului suport al masivului virtual, închiderea canalului de transfer asociat fişierului suport; d) **DIM#**; e) **OPEN . . . AS VIRTUAL**; f) **CLOSE**/obligatorie; g) **MODIFY**; h) alocări succesive/se atribuie, de regulă, zero ultimului element din ultimul masiv declarat; i) zero/şir vid; j) în linia 20 trebuie scris **FILE** 2; k) linia 80 : C1 (K%)=K%; linia 100 : **CLOSE** 2; 50 **IF** K%>29 **GO TO** 40; 60 **IF** K%=29 **GO TO** 100; Instrucţiunea de atribuire din linia 30 a fost scrisă pentru extinderea fişierului; vezi h.

☐ **Programele BASIC sînt:**

a)

```
100 REM ** CREARE FIŞIER LIVRĂRI . DAT **
110 DIM #1, C1(100), C(100), P(100)
120 PRINT
130 OPEN "LIVRĂRI . DAT" AS VIRTUAL FILE 1
140 P(100)=0
150 PRINT "VĂ ROG INTRODUCEŢI: COD BENEFICIAR/CANTITATE/PREŢ"
160 PRINT
170 INPUT "COD : " ; K%
180 IF K% > 99 GO TO 170
190 IF K%=99 GO TO 240
200 INPUT "CANTITATE : " ; C2
210 INPUT "PREŢ : " ; C3
220 C1 (K%)=K% : C(K%)=C2 : P(K%)=C3
230 GO TO 160
240 CLOSE 1
250 PRINT
260 PRINT "FIŞIERUL LIVRĂRI . DAT A FOST CREAT"
270 PRINT
280 END
```

☐ Nu răspundem.

☐ Nu răspundem.

Proiectarea și realizarea programului principal pentru crearea și întreținerea unui fișier masiv virtual de livrări (adăugare, consultare, numărare, ștergere, modificare) care apelează mai multe subrutine al căror format sursă se cunoaște. Instrucțiuni BASIC-PLUS pentru prelucrarea fișierelor masive virtuale de tip șir. Depanarea unui program. JOCURI, APLICAȚII și TESTE pentru cititor

EXEMPLUL 10-M

□ De la problemă la program

Vom aborda şi în cadrul acestei conversaţii problema creerii şi întreţinerii unui fişier masiv virtual de livrări, cu singura deosebire că structura fişierului se modifică în: denumire beneficiar, cantitate livrată şi preţ.

Funcţiunile programului sînt: creare fişier, adăugare beneficiari, consultare fişier, numărare înregistrări existente şi listare fişier, ştergere înregistrări, modificare înregistrări, oprirea execuţiei programului.

Programul are ca obiectiv ilustrarea facilităţilor limbajului BASIC-PLUS pentru prelucrarea fişierelor masive virtuale de tip şir el fiind ghidat ca şi programele conversaţiilor precedente de un meniu. Datorită faptului că dvs. aveţi deja experienţa conversaţiei cu minicalculatoarele INDEPENDENT şi CORAL vă vom prezenta, de această dată în plus, pe lîngă specificaţiile de programare (v. modulul de analiză structurată, figura 10.1),

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLUL 10 – M

Descrierea programului

Programul creează şi întreţine un fişier masiv virtual de livrări. Programul este ghidat de un meniu care permite alegerea uneia din următoarele opţiuni: creare fişier, adăugare beneficiari, consultare fişier, numărare înregistrări existente şi listare fişier, ştergere (din fişier), modificare înregistrare, sfîrşit sesiune actuală.

Intrări

Se introduc de la tastatură: codul opţiunii (un număr între 1 şi 7), numele beneficiarului, cantitatea livrată, preţ tonă benzină.

Ieşiri

Fişierul masiv virtual BENEF.DAT.

Lista de funcţiuni ale programului

1. Deschidere fişier
2. Închidere fişier
3. Afişare cap tabel BENEFICIAR/CANTITATE LIVRATĂ/PREŢ
4. Iniţializare I3
5. Căutare beneficiari înscrişi în fişier
6. Incrementare variabilă I3
7. Iniţializare variabilă indexată B$ cu şir vid
8. Iniţializare variabile indexate P, C cu zero
9. Incrementare variabilă T1
10. Afişare mesaj "FIŞIERUL EXISTĂ"
11. Testare condiţie fişier plin
12. Afişare mesaj "FIŞIER ESTE PLIN * ADĂUGĂRILE SE REFUZĂ" în caz de umplere fişier
13. Citire nume beneficiar în variabila N$
14. Iniţializare variabilă T cu zero
15. Iniţializare variabilă A cu 31
16. Identificare prima căsuţă goală
17. Iniţializare variabilă T cu 1 în caz că beneficiarul există în fişier
18. Afişare mesaj "BENEFICIARUL x x x ESTE DEJA ÎN FIŞIER"
19. Reţinere nume beneficiar dacă A<31
20. Citire cantitate în variabila C(A)
21. Citire preţ în variabila P(A)
22. Iniţializare variabilă T cu zero
23. Detectare prezenţă beneficiar solicitat
24. Testare condiţie T < > 0 şi tipărire cap tabel
25. Tipărire beneficiar, cantitate, preţ
26. Iniţializare variabilă T cu 1
27. Tipărire mesaj "A FOST ŞTERS BENEFICIARUL" atunci cînd T=1
28. Tipărire mesaj "BENEFICIAR INEXISTENT ÎN FIŞIER" atunci cînd T este diferit de 1
29. Detectare prezenţă beneficiar solicitat
30. Afişare mesaj "SPECIFICAŢI MODIFICĂRILE DORITE" atunci cînd T este diferit de zero

a)

Fig. 10.1. Modulul de analiză structurată: *a)* specificaţii de programare;

SPECIFICAŢII DE PROGRAMARE	
31. Afişare mesaj "BENEFICIARUL x x x NU EXISTĂ ÎN FIŞIER" atunci cînd T este egal cu zero	39. Specificarea opţiunii:
32. Citire nume beneficiar în variabila M$	1 Creare fişier beneficiari
33. Citire cantitate în variabila C1	2 Adăugare beneficiari
34. Citire preţ în variabila P1	3 Consultare fişier beneficiari
35. Atribuire valoare variabilă M$ lui B$ (T)	4 Numărare înregistrări existente şi listare fişier
36. Atribuire valoare variabilă C1 lui C(T)	5 Ştergere din fişier beneficiari
37. Atribuire valoare variabilă P1 lui P(T)	6 Modificare înregistrare beneficiari
38. Afişare mesaj "NOUA ÎNREGISTRARE"	7 Sfîrşit sesiune actuală
	40. Stop

a) (continuare)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M

Variabile de intrare	Variabile de stare	Variabile de ieşire
O: opţiune meniu	T1: stare fişier	B$: nume beneficiar
N$: nume beneficiar	I3: variabilă contorizare număr beneficiari	C: cantitate
M$: nume beneficiar	T: stare beneficiar	
C1: cantitate	A: locaţie goală (index)	P: preţ
P1: Preţ	B$: nume beneficiar	
	C: cantitate	
	P: preţ	
	I, P9: indecşi	

b)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (Subrutina 450)

Variabile de intrare	Variabile de stare	Variabile de ieşire
	I: variabila de control	
	B$: nume beneficiar	
	C: cantitatea	
	P: preţ	
	T1: stare fişier	

c)

Fig. 10.1. *b)* tabela de variabile; *c)* tabela de variabile (subrutina 450);

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (subrutina 570)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N$: nume beneficiar	T1: stare fișier	B$: nume beneficiar
C: cantitate	I3: variabilă contorizare număr beneficiari	C: cantitate
P: preț	T: stare beneficiar (există/nu există în fișier)	P: preț
	A: locație goală (index)	
	B$: nume beneficiar din fișier	

d)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (subrutina 800)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N$: nume beneficiar	T1: Stare fișier	B$: nume beneficiar
	T: stare beneficiar	C: cantitate
	B$: nume beneficiar (din fișier)	P: preț
	I: index	

e)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (subrutina 950)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N$: numele beneficiarului	T: stare beneficiar	N$: numele beneficiarului
	B$: nume beneficiar (din fișier)	P: preț
		C: cantitate

f)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (subrutina 1130)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N$: nume beneficiar	T: stare beneficiar	N$: nume beneficiar
M$: nume beneficiar	T1: stare fișier	B$: nume beneficiar
C1: cantitate	B$: nume beneficiar (din fișier)	C: cantitate
P1: preț		P: preț

g)

Fig. 10.1. *d)* tabela de variabile (subrutina 570); *e)* tabela de variabile (subrutina 800); *f)* tabela de variabile (subrutina 950); *g)* tabela de variabile (subrutina 1 130);

TABELA DE VARIABILE		
Nume program: EXEMPLUL 10 – M (subrutina 1350)		
Variabile de intrare	Variabile de stare	Variabile de ieşire
	T1: stare fişier I3: variabilă contorizare număr beneficiari P9: index (variabilă de control)	B$: nume beneficiar C: cantitate P: preţ

h)

TABELA DE VARIABILE		
Nume program: EXEMPLUL 10 – M (subrutina 1630)		
Variabile de intrare	Variabile de stare	Variabile de ieşire
	B$: nume beneficiar I3: Variabilă contorizare număr beneficiari	I3: variabilă contorizare număr beneficiari

i)

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE

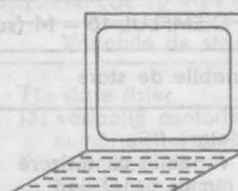
Nume program: EXEMPLUL 10 – M

Modul	Funcţiuni
DESCHID–1470	1
INIT	7, 8, 9, 10
INCHID–1530	2
NUMĂRĂ–1630	1, 4, 5, 6, 2
PREL–AD	9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
PREL–CONS	9, 22, 13, 23, 24, 25
CAP–TAB–1590	3, 4, 5, 6, 2
PREL–STERG	22, 9, 13, 7, 8, 26, 27, 28
PREL–MOD	22, 9, 13, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 25

j)

Fig. 10.1. *h)* tabela de variabile (subrutina 1 350); *i)* tabela de variabile (subrutina 1 630); *j)* alocarea funcţiunilor de prelucrare.

documentaţia de proiectare (v. modulul de proiectare structurată, fig. 10.2) şi listele cu programul sursă al subrutinelor ce intră în structura programului.



EXEMPLUL 10-M

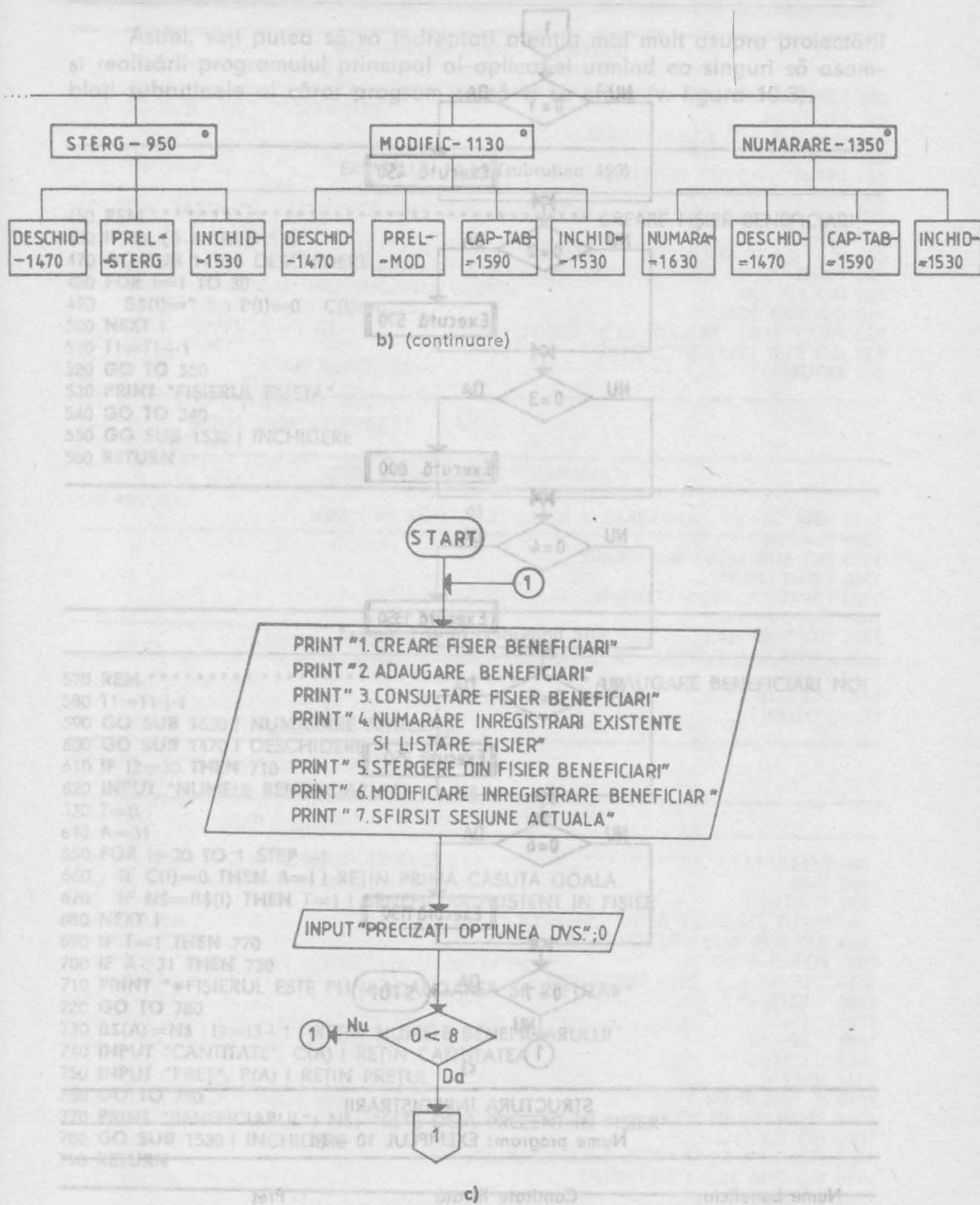


a)

- EXEMPLUL 10-M
  - MENIU
    - CREARE-450
      - DESCHID-1470
      - INIT
      - INCHID-1530
    - ADAUGARE-570
      - NUMARA-1630
      - DESCHID-1470
      - PREL-AD
      - INCHID-1530
    - CONSULTARE-800
      - DESCHID-1470
      - PREL-CONS
      - CAP-TAB-1590
      - INCHID-1530

b)

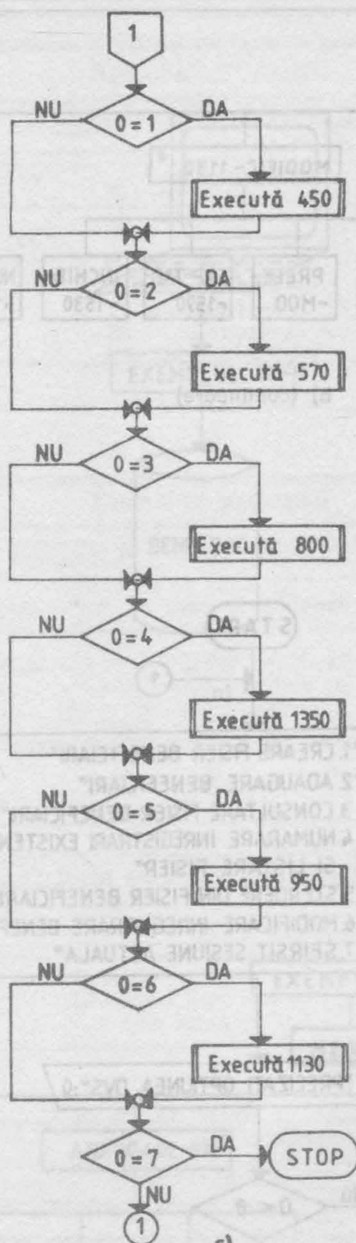
Fig. 10.2. Modulul de proiectare structurată: *a)* schema de sistem; *b)* diagrama de structură.



b) (continuare)

c)

Fig. 10.2. c) schema logică.



c)

STRUCTURA ÎNREGISTRĂRII		
Nume program: EXEMPLUL 10 – M		
Nume beneficiar	Cantitate livrată	Preţ

Fig. 10.2. *d*) structura înregistrării.

Astfel, veți putea să vă îndreptați atenția mai mult asupra proiectării și realizării programului principal al aplicației urmînd ca singuri să asamblați subrutinele al căror program sursă vi se oferă (v. figura 10.3).

EXEMPLUL 10-M (subrutina 450)

```
450 REM ************************************** CREARE FIȘIER BENEFICIARI
460 IF T1 <> 0 THEN 530
470 GO SUB 1470 ! DESCHIDERE
480 FOR I=1 TO 30
490   B$(I)=" " : P(I)=0 : C(I)=0
500 NEXT I
510 T1=T1+1
520 GO TO 550
530 PRINT "FIȘIERUL EXISTĂ"
540 GO TO 340
550 GO SUB 1530 ! ÎNCHIDERE
560 RETURN
```

a)

Exemplul 10-M (subrutina 570)

```
570 REM ************************************ ADĂUGARE BENEFICIARI NOI
580 T1=T1+1
590 GO SUB 1630 ! NUMĂRARE ARTICOLE
600 GO SUB 1470 ! DESCHIDERE
610 IF I3=30 THEN 710
620 INPUT "NUMELE BENEFICIARULUI : ", N$
630 T=0
640 A=31
650 FOR I=30 TO 1 STEP -1
660   IF C(I)=0 THEN A=I ! REȚIN PRIMA CĂSUȚĂ GOALĂ
670   IF N$=B$(I) THEN T=1 ! BENEFICIAR EXISTENT ÎN FIȘIER
680 NEXT I
690 IF T=1 THEN 770
700 IF A<31 THEN 730
710 PRINT "*FIȘIERUL ESTE PLIN*ADĂUGAREA SE REFUZĂ*"
720 GO TO 780
730 B$(A)=N$ : I3=I3+1 ! REȚIN NUMELE BENEFICIARULUI
740 INPUT "CANTITATE", C(A) ! REȚIN CANTITATEA
750 INPUT "PREȚ", P(A) ! REȚIN PREȚUL
760 GO TO 780
770 PRINT "BENEFICIARUL"; N$; "ESTE DEJA PREZENT ÎN FIȘIER"
780 GO SUB 1530 ! ÎNCHIDERE
790 RETURN
```

b)

Fig. 10.3. Programele sursă ale subrutinelor; *a)* subrutina 450; *b)* subrutina 570;

EXEMPLUL 10-M (subrutina 800)

```
800 REM *************************** CONSULTARE FIȘIER BENEFICIARI
810 T1=T1+1
820 GO SUB 1470 ! DESCHIDERE
830 T=0
840 INPUT "NUMELE BENEFICIARULUI : ", N$
850 FOR I=1 TO 30
860   IF N$=B$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
870 NEXT I
880 IF T <> 0 THEN 910
890 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ ÎN FIȘIER"
900 GO TO 930
910 GO SUB 1590
920 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
930 GO SUB 1530 ! ÎNCHIDERE
940 RETURN
```

c)

EXEMPLUL 10-M (subrutina 1350)

```
1350 REM SRT DE NUMĂRARE A ÎNREG. EXISTENTE ÎN FIȘIER
1360 T1=T1+1
1370 GO SUB 1630 ! NUMĂRARE
1380 PRINT : PRINT
1390 PRINT "FIȘIERUL CONȚINE"; I3; "ÎNREGISTRĂRI"
1400 GO SUB 1470
1410 GO SUB 1590
1420 PRINT B$(P9), C(P9), P(P9) FOR P9=1 TO I3
1430 PRINT
1440 GO SUB 1530
1450 RETURN
```

d)

EXEMPLUL 10-M (subrutina 950)

```
950 REM **************************** ȘTERGERE BENEFICIAR DIN FIȘIER
950 T=0
970 T1=T1+1
980 INPUT "NUMELE BENEFICIARULUI : ", N$
990 GO SUB 1470 ! DESCHIDERE
1000 FOR I=1 TO 30
1010   IF N$ <> B$(I) THEN 1060
1020   B$(I)=" "
1030   T=1
1040   P(I)=0
1050   C(I)=0
1060 NEXT I
1070 IF T=1 THEN 1100
1080 PRINT "BENEFICIAR INEXISTENT ÎN FIȘIER"
1090 GO TO 1110
1100 PRINT "A FOST ȘTERS BENEFICIARUL"; N$
1110 GO SUB 1530 ! ÎNCHIDERE
1120 RETURN
```

e)

Fig. 10.3. *c)* subrutina 800; *d)* subrutina 1 350; *e)* subrutina 950;

EXEMPLUL 10-M (subrutina 1130)

```
1130 REM ****************************** MODIFICARE ÎNREGISTRARE BENEFICIAR
1140 GO SUB 1470 ! DESCHIDERE
1150 T=0 : T1=T1+1
1160 INPUT "NUMELE BENEFICIARULUI : ", N$
1170 FOR I=1 TO 30
1180 IF N$=B$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
1190 NEXT I
1200 IF T <> 0 THEN 1230
1210 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ ÎN FIŞIER"
1220 GO TO 1330
1230 PRINT "SPECIFICAŢI MODIFICĂRILE DORITE"
1240 INPUT "NUME BENEFICIAR", M$
1250 INPUT "CANTITATE", C1
1260 INPUT "PREŢ", P1
1270 IF M$ <> " " THEN B$(T)=M$
1280 IF C1 <> 0 THEN C(T)=C1
1290 IF P1 <> 0 THEN P(T)=P1
1300 PRINT "NOUĂ ÎNREGISTRARE: "
1310 GO SUB 1590
1320 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
1330 GO SUB 1530 ! ÎNCHIDERE
1340 RETURN
```

f)

EXEMPLUL 10-M (subrutinele 1470, 1530, 1590, 1630)

```
1460 REM SUBRUTINE
1470 REM DESCHIDERE FIŞIER
1480 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1490 PRINT
1500 PRINT " ****ÎNCEPUT FAZĂ****"
1510 PRINT
1520 RETURN
1530 REM ÎNCHIDERE FIŞIER
1540 CLOSE 1
1550 PRINT
1560 PRINT " ****TERMINAT FAZA ****"
1570 PRINT
1580 RETURN
1590 REM CAP TABEL
1600 PRINT "BENEFICIAR /CANTITATE LIVRATĂ/ PREŢ"
1610 PRINT " **********************************": PRINT
1620 RETURN
1630 REM SUBRUTINĂ NUMĂRARE BENEFICIARI ÎNSCRIŞI ÎN FIŞIER
1640 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1650 I3=0
1660 FOR I=1 TO 30
1670   IF B$(I)=" " THEN 1690
1680   I3=I3+1
1690 NEXT I
1700 CLOSE 1
1710 RETURN
```

g)

Fig. 10.3. *f)* subrutina 1 130; *g)* subrutinele 1 470, 1 530, 1 590, 1 630.

□ Codificarea în limbajul BASIC-PLUS

vol. 2, pag. 236

Programul conversației

Programul principal al conversației cuprinde liniile 100–440.

```
100 REM PROGRAMUL CREAZĂ ŞI ÎNTREŢINE UN FIŞIER DE BENEFICIARI
110 REM FUNCŢIUNILE PROGRAMULUI:
120 REM – CREARE FIŞIER BENEFICIARI
130 REM – ADĂUGARE BENEFICIARI
140 REM – CONSULTARE FIŞIER BENEFICIARI
150 REM – ŞTERGERE BENEFICIARI
160 REM – LISTARE BENEFICIARI
170 REM – MODIFICARE BENEFICIAR EXISTENT
180 REM DEFINIREA FIŞIERULUI VIRTUAL
190 DIM #1, B$(30)=40, C(30), P(30), B1$(30, 30)
200 REM B$ – NUME BENEFICIAR
210 REM C – CANTITATE
220 REM P – PREŢ
230 PRINT
240 PRINT " 1. CREARE FIŞIER BENEFICIARI"
250 PRINT " 2. ADĂUGARE BENEFICIARI"
260 PRINT " 3. CONSULTARE FIŞIER BENEFICIARI"
270 PRINT " 4. NUMĂRARE ÎNREGISTRĂRI EXISTENTE ŞI LISTARE FIŞIER"
280 PRINT " 5. ŞTERGERE DIN FIŞIER BENEFICIARI"
290 PRINT " 6. MODIFICARE ÎNREGISTRARE BENEFICIAR"
300 PRINT " 7. SFÎRŞIT SESIUNE ACTUALĂ"
310 PRINT
320 INPUT "PRECIZAŢI OPŢIUNEA DVS."; O
330 O<8 THEN 360
340 PRINT "OPŢIUNE ERONATĂ"
350 GO TO 230
360 IF O=1 THEN GO SUB 450
370 IF O=2 THEN GO SUB 570
380 IF O=3 THEN GO SUB 800
390 IF O=4 THEN GO SUB 1350
400 IF O=5 THEN GO SUB 950
410 IF O=6 THEN GO SUB 1130
420 IF O=7 THEN GO TO 1720
430 GO TO 230
440 PRINT
```

După ce în liniile 240–300 se listează meniul (cu cele şapte opţiuni), în linia 320 programul citeşte dinamic opţiunea utilizatorului. În cazul cînd pentru variabila de intrare O se introduce o valoare mai mare sau egală cu 8, se afişează mesajul "OPŢIUNE ERONATĂ" şi ciclul se reia. Funcţie de opţiunea aleasă programul selectează una din subrutinele: 450, 570, 800, 1350, 950, 1130 pe care vă invităm să le analizaţi în detaliu.

TEST

Trasaţi schemele logice ale subrutinelor din program.

În cele ce urmează ne vom opri numai asupra posibilităţilor limbajului BASIC-PLUS privind prelucrarea fişierelor masive virtuale de tip şir.

În linia 190 a programului principal am declarat numărul virtual de tip şir B$ (numele beneficiarului).

```
190 DIM # 1, B$(30)=40, C(30), P(30), B1$(30, 30)
```

În limbajul BASIC-PLUS şirurile-elemente ale unui masiv virtual de tip şir (nenumeric) au aceeaşi lungime fixă, declarată prin **DIM#** (v. linia 190).

Dacă această lungime declarată (40) nu este o putere a lui 2, cuprinsă între 2 şi 512, ea este implicit transformată la puterea lui 2 imediat superioară. Astfel, linia 190 declară masivul virtual B$ ca avînd elemente de lungime maximă 64, deoarece 32<40<64.

Formatul general al instrucţiunii **DIM#** pentru declararea masivelor virtuale de tip şir este:

Format general
DIM# ⟨expresie numerică⟩ ⟨listă⟩

în care: ⟨expresie numerică⟩ are semnificaţia ştiută (v. conversaţia 9); ⟨listă⟩ reprezintă o listă de elemente ce precizează numele masivelor şi dimensiunilor acestora.

Pentru masivele şir, elementele pot fi de forma:

Format general
⟨masiv⟩(⟨dim 1⟩ [,⟨dim 2⟩]) [=⟨constantă întreagă⟩|⟨variabila întreagă⟩

unde:

⟨masiv⟩ reprezintă un nume de variabilă indexată;

⟨dim 1⟩, ⟨dim 2⟩ reprezintă dimensiunile declarate ale masivelor; poate fi orice constantă sau variabilă simplă întreagă;

⟨constantă întreagă⟩ şi ⟨variabilă întreagă⟩ reprezintă lungimea maximă admisă pentru elementele masivului şir.

Reguli

- Un element aparţinînd unui masiv şir nu poate depăşi lungimea declarată prin **DIM #**.
- Un element al unui masiv şir poate avea o lungime mai mică decît cea declarată prin **DIM #**, în acest caz făcîndu-se alinierea sa la stînga şi completîndu-se spaţiul disponibil cu caracterul nul (cod ASCII=0).
- Dacă **DIM #** nu conţine declaraţia de lungime, aceasta se consideră implicit egală cu 16.
- Pentru referirea unui masiv creat într-un alt program trebuie să fie menţionată corect poziţia relativă a acestui masiv faţă de începutul fişierului (cazul masivului B1$ declarat în linia 190; acesta va fi folosit în programul din conversaţia următoare).

Aplicaţie. Să se realizeze specificaţiile de programare, documentaţia de proiectare şi programul BASIC pentru crearea, consultarea, actualizarea şi listarea unui fişier masiv virtual de profiluri standardizate.

Profilurile sînt bare metalice cu secţiunea transversală constantă şi cu o anumită formă standardizată sau prescrisă printr-o normă. Profilurile se folosesc în construcţii metalice civile, în construcţii de coloane tubulare şi alte utilaje, în construcţii de nave, de material rulant, de autovehicule etc.

Din punct de vedere al procedeelor de fabricare se deosebesc următoarele tipuri de profiluri: profiluri fasonate, laminate la cald din oţeluri; profiluri fasonate formate la rece din bandă, profiluri simple laminate la cald din oţeluri, profiluri simple calibrate etc.

În tabelul 10.1 se prezintă mai multe tipuri de profiluri standardizate cărora li se precizează: simbolul, denumirea, tipul (structurat în cod şi denumire), secţiunea (codurile

Tabelul 10.1

Simbol profil	Denumire profil	Tip profil Cod	Tip profil Denumire	Secțiune	Simbol general	Dimensiuni secțiune	Lungime de fabricație	STAS
1	2	3	4	5	6	7	8	9
L	oțel cornier cu aripi egale	1	fasonate, laminate la cald din oțel	1	axaxg	a= 20 ... 160 g= 3 ... 18	3 000 ... 12 000	424-80
LL	oțel cornier cu aripi neegale	1	fasonate, laminate la cald din oțel	2	axbxg	a= 30 ... 150 b= 20 ... 100 g= 3 ... 134	4 000 ... 12 000	425-80
I	oțel I	1	fasonate, laminate la cald din oțel	3	Ih	h= 80 ... 400 b= 42 ... 155 g= 3,9 ... 14,4	5 000 ... 15 000 (6 000 ... 15 000)	565-80
IE	oțel I economic	1	fasonate, laminate la cald din oțel	3	IEh	h=100 ... 400 b= 55 ... 155 g= 4,5 ... 8,0	8 000 ... 15 000 (6 000 ... 15 000)	7550-77
T	oțel cu aripi egale și muchii rotunjite	1	fasonate, laminate la cald din oțel	4	Th	h= 20 ... 50 a=h=20 ... 50 g= 3 ... 6	4 000 ... 8 000	566-80
U	oțel U	1	fasonate, laminate la cald din oțel	5	Uh	h= 65 ... 300 b= 42 ... 100 g= 5,5 ... 10,0	5 000 ... 15 000 (6 000 ... 15 000)	564-80
UE	oțel U economic	1	fasonate, laminate la cald din oțel	5	UEh	h= 50 ... 300 b= 32 ... 100 g= 4,4 ... 6,5	8 000 ... 15 000 (6 000 ... 15 000)	7551-77

LI	profil cornier cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	axaxg	a= 16 ... 125 g= 1 ... 5	4 000 ... 8 000	7836-80
LLI	profil cornier au aripi neegale	2	fasonate, formate la rece din bandă de oțel	0	axbxg		4 000 ... 8 000	8250-80
TI	profil T cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	axhxg	h=30 și 40 a=h g=2	4 000 ... 8 000	8249-80
UI	profil U cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	hxbxg	h=16 ... 200 b=10 ... 100 g= 1,5 ... 5,0	4 000 ... 8 000	7835-80
UNI	profil U cu aripi neegale	2	fasonate, formate la rece din bandă de oțel	0	hxBxbxg	h=20 ... 140 B=20 ... 100 b=16 ... 60 g= 1,5 ... 5,0	4 000 ... 8 000	8610-80
ZI	profil Z cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	hxbxg	h=20 ... 100 b=16 ... 50 g= 1,5 ... 4,0	4 000 ... 8 000	8296-80
ZNI	profil Z cu aripi neegale	2	fasonate, formate la rece din bandă de oțel	0	hxBxbxg	h=25 ... 120 B=20 ... 50 b=16 ... 40 g= 1,5 ... 4,0	4 000 ... 8 000	8609-80

Tabelul 10.1 (continuare)

1	2	3	4	5	6	7	8	9
ΩI	profil omogen cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	hxbxaxg	h=20 . . . 100 b=20 . . . 50 a=10 . . . 32 g= 1 . . . 3	4 000 . . . 8 000	8367-80
LT	oțel lat	3	simple, laminate la cald sau trase la rece din oțeluri	6	axb	a=20 . . . 150 b= 5 . . . 50	3 000 . . . 7 000 (1 500 . . . 7 000)	395-80
LTC	oțel lat calibrat	3	simple, laminate la cald sau trase la rece din oțeluri	6	axb	a=18 . . . 50 b= 5 . . . 16	2 000 . . . 6 000 (1 500 . . . 6 000)	6972-80
4L	oțel pătrat	3	simple, laminate la cald sau trase la rece din oțeluri	7	4La	a=8 . . . 60	3 000 . . . 9 000 (1 500 . . . 9 000)	334-80
4LC	oțel pătrat calibrat	3	simple, laminate la cald sau trase la rece din oțeluri	7	Ta	a=3,5 . . . 50,0	1 500 . . . 6 000 (500 . . . 6 000)	6554-80

6L	oțel hexagonal	3	simple, laminate la cald sau trase la rece din oțeluri	8	6LS	S=8 . . . 72	2 000 . . . 6 000	7828-78
TR	oțel hexagonal calibrat	3	simple, laminate la cald sau trase la rece din oțeluri	8	TRS	S=4 . . . 70	1 500 . . . 6 000	2305-80
Ø	oțel rotund	3	simple, laminate la cald sau trase la rece din oțeluri	9	Ød	d=10 . . . 60	3 000 . . . 10 000	333-80
TS	oțel rotund calibrat	3	simple, laminate la cald sau trase la rece din oțeluri	9	TSd	d=2 . . . 70	1 500 . . . 6 000	1800-80

secțiunii corespund figurii 10.4), simbolul general, dimensiunile secțiunii, lungimea de fabricație și STAS-ul.

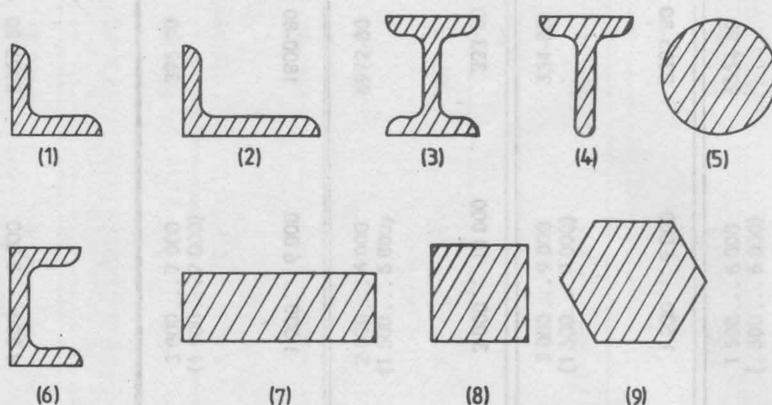
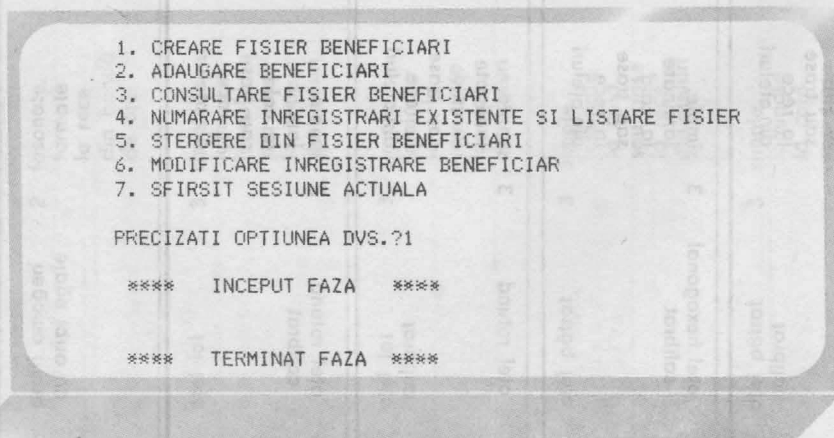


Fig. 10.4. Profiluri standardizate.

Observație. Se recomandă ca programul BASIC–PLUS pe care-l realizați să fie ghidat de către un meniu.

Rezultatele opțiunilor

Rezultatele conversației cu minicalculatoarele INDEPENDENT și CORAL prin intermediul programului EXEMPLUL 10-M sînt ilustrate în figura 10.5.



```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?1

 ****   INCEPUT FAZA   ****


 ****   TERMINAT FAZA  ****
```

a)

Fig. 10.5. Opțiuni: *a)* opțiunea 1;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PECO TITAN
CANTITATE ?1000
PRET      ?7500

 ****   TERMINAT FAZA  ****
```

b)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PECO MILITARI
CANTITATE ?30000
PRET      ?7500

 ****   TERMINAT FAZA  ****
```

c)

Fig. 10.5. *b)* opţiunea 2; *c)* opţiunea 2;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE  2  INREGISTRARI

 ****   INCEPUT FAZA    ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
======================================

PECO TITAN       1000          7500
PECO MILITARI  30000           7500

 ****   TERMINAT FAZA  ****
```

d)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PECO-COLENTINA
CANTITATE ?1000
PRET       ?7100

 ****   TERMINAT FAZA  ****
```

e)

Fig. 10.5. *d)* opțiunea 4; *e)* opțiunea 2;

```
 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

 PRECIZATI OPTIUNEA DVS.?1
FISIERUL EXISTA
OPTIUNE ERONATA

 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

 PRECIZATI OPTIUNEA DVS.?2

  ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PECO-DR.TABEREI
CANTITATE ?3000
PRET      ?8000

  ****   TERMINAT FAZA  ****
```

f)

```
 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

 PRECIZATI OPTIUNEA DVS.?2

  ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?CRAIOVA-VALEA ROSIE
CANTITATE ?4000
PRET      ?6000

  ****   TERMINAT FAZA  ****
```

g)

Fig. 10.5. *f)* opţiunea 1; *g)* opţiunea 2;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PLOIESTI-BARIERA
CANTITATE ?3000
PRET      ?7000

 ****   TERMINAT FAZA  ****
```

h)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?BUZAU-CENTRU
CANTITATE ?3000
PRET      ?7800

 ****   TERMINAT FAZA  ****
```

i)

Fig. 10.5. *h)* opțiunea 2; *i)* opțiunea 2;

```
 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

 PRECIZATI OPTIUNEA DVS.?2

  ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?IASI-COPOU
CANTITATE ?300
PRET      ?7890

  ****   TERMINAT FAZA  ****
```

j)

```
 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

 PRECIZATI OPTIUNEA DVS.?2

  ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?URZICENI
CANTITATE ?1000
PRET      ?7000

  ****   TERMINAT FAZA  ****
```

k)

Fig. 10.5. *l)* opțiunea 2; *k)* opțiunea 2;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4


FISIERUL CONTINE 9 INREGISTRARI

 ****   INCEPUT FAZA   ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
=======================================

PECO TITAN       1000          7500
PECO MILITARI   30000          7500
PECO-COLENTINA                 1000        7100
PECO-DR.TABEREI                3000        8000
CRAIOVA-VALEA ROSIE            4000        6000
PLOIESTI-BARIERA               3000        7000
BUZAU-CENTRU     3000          7800
IASI-COPOU        300          7890
URZICENI         1000          7000

 ****   TERMINAT FAZA   ****
```

l)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA
PRECIZATI OPTIUNEA DVS.?6

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?CRAIOVA-VALEA ROSIE
SPECIFICATI MODIFICARILE DORITE
NUME BENEFICIAR ?CRAIOVA
CANTITATE      ?4000
PRET           ?7800
NOUA INREGISTRARE :
BENEFICIAR  /CANTITATE LIVRATA / PRET
=======================================

CRAIOVA                  4000      7800

 ****   TERMINAT FAZA   ****
```

m)

Fig. 10.5. *l)* opţiunea 4; *m)* opţiunea 6;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE  9  INREGISTRARI

 ****   INCEPUT FAZA   ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
=====================================

PECO TITAN        1000           7500
PECO MILITARI  30000             7500
PECO-COLENTINA                   1000          7100
PECO-DR.TABEREI                  3000          8000

CRAIOVA           4000           7800
PLOIESTI-BARIERA                 3000          7000
BUZAU-CENTRU   3000              7800
IASI-COPOU        300            7890
URZICENI          1000           7000

 ****   TERMINAT FAZA  ****
```

n)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?6

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PECO-COLENTINA
SPECIFICATI MODIFICARILE DORITE
NUME BENEFICIAR ?PECO-CV
CANTITATE        ?6789
PRET             ?7800
NOUA INREGISTRARE :
BENEFICIAR  /CANTITATE LIVRATA / PRET
=====================================

PECO-CV              6789      7800

 ****   TERMINAT FAZA  ****
```

o)

Fig. 10.5. *n)* opţiunea 4; *o)* opţiunea 6;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?6

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PLOIESTI-BARIERA
SPECIFICATI MODIFICARILE DORITE
NUME BENEFICIAR ?PLOIESTI
CANTITATE       ?5655
PRET            ?7600
NOUA INREGISTRARE :
BENEFICIAR  /CANTITATE LIVRATA / PRET
=====================================

PLOIESTI              5655      7600

 ****   TERMINAT FAZA   ****
```

p)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE  9  INREGISTRARI

 ****   INCEPUT FAZA   ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
=====================================

PECO TITAN      1000          7500
PECO MILITARI   30000         7500
PECO-CV         6789          7800
PECO-DR.TABEREI               3000           8000
CRAIOVA         4000          7800
PLOIESTI        5655          7600
BUZAU-CENTRU    3000          7800
IASI-COPOU      300           7890
URZICENI        1000          7000

 ****   TERMINAT FAZA   ****
```

r)

Fig. 10.5. *p)* opţiunea 6; *r)* opţiunea 4;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?5
NUMELE BENEFICIARULUI : ?ICITPR-PLOIESTI

****  INCEPUT FAZA  ****

BENEFICIAR INEXISTENT IN FISIER

****  TERMINAT FAZA  ****
```

s)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?5
NUMELE BENEFICIARULUI : ?PECO-DR.TABEREI

****  INCEPUT FAZA  ****

A FOST STERS BENEFICIARUL PECO-DR.TABEREI

****  TERMINAT FAZA  ****
```

t)

Fig. 10.5. *s)* opțiunea 5; *t)* opțiunea 5;

```
 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE  8  INREGISTRARI

 ****   INCEPUT FAZA   ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
=====================================

PECO TITAN        1000        7500
PECO MILITARI     30000       7500
PECO-CV           6789        7800
                  0           0
CRAIOVA           4000        7800
PLOIESTI          5655        7600
BUZAU-CENTRU      3000        7800
IASI-COPOU        300         7890

 ****   TERMINAT FAZA   ****
```

u)

```
 1. CREARE FISIER BENEFICIARI
 2. ADAUGARE BENEFICIARI
 3. CONSULTARE FISIER BENEFICIARI
 4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
 5. STERGERE DIN FISIER BENEFICIARI
 6. MODIFICARE INREGISTRARE BENEFICIAR
 7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?SUCEAVA
CANTITATE ?5600
PRET     ?7800

 ****   TERMINAT FAZA   ****
```

v)

Fig. 10.5. *u)* opțiunea 4; *v)* opțiunea 2;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE  9  INREGISTRARI

 ****   INCEPUT FAZA   ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
=====================================

PECO TITAN      1000          7500
PECO MILITARI   30000         7500
PECO-CV         6789          7800
SUCEAVA         5600          7800
CRAIOVA         4000          7800
PLOIESTI        5655          7600
BUZAU-CENTRU    3000          7800
IASI-COPOU      300           7890
URZICENI        1000          7000

 ****   TERMINAT FAZA   ****
```

z)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?TITU
CANTITATE ?5600
PRET      ?7600

 ****   TERMINAT FAZA   ****
```

w1)

Fig. 10.5. *z)* opțiunea 4; *w.1)* opțiunea 4;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?CERNAVODA
CANTITATE ?5400
PRET    ?6000

****   TERMINAT FAZA   ****
```

w2)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?ARAD
CANTITATE ?5555
PRET    ?7780

****   TERMINAT FAZA   ****
```

w3)

Fig. 10.5. *w.2)* opţiunea 2; *w.3)* opţiunea 2;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?ALBA-IULIA
CANTITATE ?500
PRET      ?7914

 ****   TERMINAT FAZA  ****
```

w4)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?VALENI
CANTITATE ?1000
PRET      ?7400

 ****   TERMINAT FAZA  ****
```

w5)

Fig. 10.5. *w.4)* opţiunea 2; *w.5)* opţiunea 2;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE  14  INREGISTRARI

 ****   INCEPUT FAZA   ****

BENEFICIAR  /CANTITATE LIVRATA / PRET
=======================================

PECO TITAN      1000       7500
PECO MILITARI   30000      7500
PECO-CV         6789       7800
SUCEAVA         5600       7800
CRAIOVA         4000       7800
PLOIESTI        5655       7600
BUZAU-CENTRU    3000       7800
IASI-COPOU      300        7890
URZICENI        1000       7000
TITU            5600       7600
CERNAVODA       5400       6000
ARAD            5555       7780
ALBA-IULIA      500        7914
VALENI          1000       7400

 ****   TERMINAT FAZA  ****
```

w6)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?3

 ****   INCEPUT FAZA   ****

NUMELE BENEFICIARULUI : ?PLOIESTI
BENEFICIAR  /CANTITATE LIVRATA / PRET
=======================================

PLOIESTI            5655       7600

 ****   TERMINAT FAZA  ****
```

w7)

Fig. 10.5. *w.6)* opțiunea 4; *w.7)* opțiunea 3;

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?3

 ****  INCEPUT FAZA  ****

NUMELE BENEFICIARULUI : ?PRAHOVA
BENEFICIARUL PRAHOVA NU EXISTA IN FISIER

 ****  TERMINAT FAZA  ****
```

w8)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?3

 ****  INCEPUT FAZA  ****

NUMELE BENEFICIARULUI : ?COMBINATUL PETROCHIMIC VALEA CALUGAREASCA
BENEFICIARUL COMBINATUL PETROCHIMIC VALEA CALUGAREASCA
NU EXISTA IN FISIER

 ****  TERMINAT FAZA  ****
```

w9)

```
1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?7
STOP AT LINE 1720
```

w10)

Fig. 10.5. *w.8)* opțiunea 3; *w.9)* opțiunea 3; *w.10)* opțiunea 7.

Joc pe COMMODORE

```
10 REM JOCUL BILELOR
20 PRINT CHR$ (147)
30 LO=1024 : HI=2023 : WI=40 :
   D=54272
40 L=HI-LO : TK=81
50 INPUT "CÎTE BILE (PÎNĂ LA 9)"; N
60 DIM S (N, 10, 2), T(N), P(N), C(N)
70 FOR I=1 TO N : P(I)=INT
   (RND(1) * L)+LO
80 PRINT
90 PRINT "CÎTE MIŞCĂRI PENTRU
   BILA #"; I
100 INPUT T(I)
110 FOR J=1 TO T(I)
120 PRINT
130 PRINT "INTRODUCEŢI H ŞI V PEN-
    TRU BILĂ"; I; "MUTA M INTROD"; J
140 INPUT S(I, J, 1), S(I, J, 2)
150 NEXT J : NEXT I
160 PRINT CHR$ (147)
170 FOR I=1 TO N : C(I)=C(I)+1
180 IF C(I)>T(I) THEN C(I)=1
190 NP=P(I)+S(I, C(I), 1)+WI *
    S(I, C(I), 2)
200 IF NP>HI THEN NP=NP-L
210 IF NP<LO THEN NP=NP+L
220 POKE NP, TK : POKE P(I), 32:
    POKE NP+D, 6+I
230 P(I)=NP
240 NEXT I
250 GOTO 170
```

TEMA 10

☐ Răspundeţi prin DA sau NU la următoarele întrebări:

– Lungimea unui element de masiv şir este cuprinsă între 2 şi 512.

– Linia de program

```
150 DIM #2, XS (35)
```

declară masivul X$ ca avînd elemente de lungime maximă 64.

– Elementul unui masiv şir nu poate depăşi lungimea declarată prin DIM#.

– Linia de program

```
30 DIM # 3, A$(20)=4
```

declară masivul A$ cu 20 elemente, fiecare element avînd lungimea maximă de 4 octeţi.

☐ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Linia de program

```
10 DIM #1, B$ (10)=70
```

declară masivul .... cu .... elemente, fiecare element avînd lungimea maximă de .... octeţi.

b) Linia de program

```
30 DIM #7, N$ (60)
```

declară masivul .... cu .... elemente, fiecare element avînd lungimea maximă de .... octeţi.

c) Linia de program

```
10 DIM #6, U$ (30)=8
```

declară masivul .... cu .... elemente, fiecare element avînd lungimea maximă de .... octeți.

d) Linia de program

```
10 DIM # 1, C$ (?)
```

declară masivul C$ cu .... elemente, fiecare element avînd lungimea maximă de 16 octeți.

☐ Scrieți cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Se consideră fișierul masiv virtual ACHIZIȚII care conține pentru fiecare articol utilizat în activitatea de întreținere a utilajelor dintr-o întreprindere constructoare de mașini, denumirea sa și prețurile de achiziție din țară, est și vest. Să se creeze și să se întrețină acest fișier utilizînd subrutinele descrise în EXEMPLUL 10-M.

b) Se consideră fișierul masiv virtual PROIECTE care conține pentru fiecare articol (proiect) codul, denumirea și caracteristicile tehnice ale produsului proiectat. Să se creeze și să se întrețină acest fișier utilizînd subrutinele de adăugare, consultare, numărare, ștergere și modificare prezentate în Conversația 10.

c) În programul de mai jos (EXEMPLUL 10-M, versiunea 1) s-au strecurat mai multe greșeli de logică. Identificați aceste erori și realizați o versiune corectă a programului.

```
100 REM PROGRAMUL CREAZĂ ȘI ÎNTREȚINE UN FIȘIER DE BENEFICIARI
110 REM FUNCȚIUNILE PROGRAMULUI
120 REM – CREARE FIȘIER BENEFICIARI
130 REM – ADĂUGARE BENEFICIARI
140 REM – CONSULTARE FIȘIER BENEFICIARI
150 REM – ȘTERGERE BENEFICIARI
160 REM – LISTARE BENEFICIARI
170 REM – MODIFICARE BENEFICIAR EXISTENT
180 REM DEFINIREA FIȘIERULUI VIRTUAL
190 DIM #1, B$(30)=40, C(30), P(30), B1$(30, 30)
200 REM B$ – NUME BENEFICIAR
210 REM C – CANTITATE
220 REM P – PREȚ
230 PRINT
240 PRINT "1. CREARE FIȘIER BENEFICIARI"
250 PRINT "2. ADĂUGARE BENEFICIARI"
260 PRINT "3. CONSULTARE FIȘIER BENEFICIARI"
270 PRINT "4. NUMĂRARE ÎNREGISTRĂRI EXISTENTE ȘI LISTARE FIȘIER"
280 PRINT "5. ȘTERGERE DIN FIȘIER BENEFICIARI"
290 PRINT "6. MODIFICARE ÎNREGISTRARE BENEFICIAR"
300 PRINT "7. SFÎRȘIT SESIUNE ACTUALĂ"
310 PRINT
320 INPUT "PRECIZAȚI OPȚIUNEA DVS."; 0
330 IF 0<8 THEN 350
340 PRINT "OPȚIUNE ERONATĂ" : GO TO 230
350 IF 0=1 THEN GO SUB 440
```

```
360 IF 0=2 THEN GO SUB 540
370 IF 0=3 THEN GO SUB 760
380 IF 0=4 THEN GO SUB 1260
390 IF 0=5 THEN GO SUB 900
400 IF 0=6 THEN GO SUB 1030
410 IF 0=7 THEN GO TO 1630
420 GO TO 230
430 PRINT
440 REM ++++++++++++ CREARE FIŞIER BENEFICIARI
450 IF T1 < > O THEN 510
460 GO SUB 1380 ! DESCHIDERE
470 FOR I=1 TO 30
480 B$(I)="" : P(I)=0: C(I)=0
490 NEXT I
500 T1=T1+1 : GO TO 340
510 PRINT "FIŞIERUL EXISTĂ" : GOTO 350
520 GOSUB 1440
530 RETURN
540 REM ++++++++++++ ADĂUGARE BENEFICIARI NOI
550 T1=T1+1
560 GO SUB 1540 ! NUMĂRARE ARTICOLE
570 GO SUB 1380 ! DESCHIDERE
580 IF I3=30 THEN 680
590 INPUT "NUMELE BENEFICIARULUI : " ; N$
600 T=O : A=0
610 FOR I=30 TO 1 STEP -1
620 IF C(I)=0 THEN A=I ! REŢIN PRIMA CĂSUŢĂ GOALĂ
630 IF N$=B$(I) THEN T=1 BENEFICIAR EXISTENT ÎN FIŞIER
640 PRINT "A="; A, "I=", I; B$(I); TAB(60); N$; C(I); P(I)
650 NEXT I
660 IF T=1 THEN 730
670 IF A< 31 THEN 690
680 PRINT " * FIŞIERUL ESTE PLIN * ADĂUGĂRILE SE REFUZĂ * " : GO TO 520
690 B$(A)=N$ : I3=I3+1 ! REŢIN NUMELE BENEFICIARULUI
700 INPUT "CANTITATE", C(A) ! REŢIN CANTITATEA
710 INPUT "PREŢ", P(A) ! REŢIN PREŢUL
720 GO TO 740
730 PRINT "BENEFICIARUL"; N$"; "ESTE DEJA PREZENT IN FIŞIER"
740 GO SUB 1440 ! ÎNCHIDERE
750 RETURN
760 REM ++++++++++++ CONSULTARE FIŞIER BENEFICIARI
770 T1=T1+1
780 GO SUB 1380 ! DESCHIDERE
790 T=0
800 INPUT "NUMELE BENEFICIARULUI : ", N$
810 FOR I=1 TO 30
820 IF N$=B$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
830 NEXT I
840 IF T < > O THEN 860
850 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ ÎN FIŞIER": GO TO 880
860 GO SUB 1500
870 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
880 GO SUB 1440 ! ÎNCHIDERE
890 RETURN
900 REM ++++++++++++ ŞTERGERE BENEFICIAR DIN FIŞIER
910 T=O : T1=T1+1
920 INPUT "NUMELE BENEFICIARULUI : " , N$
930 GO SUB 1380 ! DESCHIDERE
940 FOR I=1 TO 30
950 IF N$ < > B$(I) THEN 970
960 B$(I)=" " : C(I)=O : P(I)=O : T=1
970 NEXT 1
980 IF T=1 THEN 1000
```

```
990 PRINT "BENEFICIAR INEXISTENT ÎN FIŞIER" : GO TO 638
1000 PRINT "A FOST ŞTERS BENEFICIARUL"; N$
1010 GO SUB 1440 ! ÎNCHIDERE
1020 RETURN
1030 REM ++++++++++++ MODIFICARE ÎNREGISTRARE BENEFICIAR
1040 GO SUB 1380 ! DESCHIDERE
1050 T=O : T1=T1+1
1060 INPUT "NUMELE BENEFICIARULUI : ", N$
1070 FOR I=1 TO 30
1080 IF N$=B$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
1090 NEXT I
1100 IF T <> O THEN 1140
1110 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ ÎN FIŞIER" : GO TO 1230
1120 GO SUB 1500
1130 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
1140 PRINT "SPECIFICAŢI MODIFICĂRILE DORITE"
1150 INPUT "NUME BENEFICIAR"; M$
1160 INPUT "CANTITATE", C1
1170 INPUT "PREŢ", P1
1180 IF M$ <> " " THEN B$(T)=M$
1390 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1200 IF P1 <> O THEN P(T)=P1
1210 PRINT "NOUA ÎNREGISTRARE : "
1220 GO SUB 1500
1230 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
1240 GO SUB 1440 ! ÎNCHIDERE
1250 RETURN
1260 REM SRT DE NUMĂRARE A ÎNREG. EXISTENTE ÎN FIŞIER
1270 T1=T1+1
1280 GO SUB 1540 ! NUMĂRARE
1290 PRINT : PRINT
1300 PRINT "FIŞIERUL CONŢINE"; I3; "ÎNREGISTRĂRI"
1310 GO SUB 1380
1320 GO SUB 1500
1330 PRINT B$(P9), C(P9), P(P9) FOR P9=1 TO I3
1340 PRINT
1350 GO SUB 1440
1360 RETURN
1370 REM SUBRUTINE
1380 REM DESCHIDERE FIŞIER
1390 OPEN "BENEF.DAT" AS VIRTUAL FILE 1
1400 PRINT
1410 PRINT "**** ÎNCEPUT FAZA                    **** "
1420 PRINT
1430 RETURN
1440 REM ÎNCHIDERE FIŞIER
1450 CLOSE 1
1460 PRINT
1470 PRINT "**** TERMINAT FAZA                    **** "
1480 PRINT
1490 RETURN
1500 REM CAP TABEL
1510 PRINT "BENEFICIAR/CANTITATE LIVRATĂ/PREŢ"
1520 PRINT "==============" : PRINT
1530 RETURN
1540 REM SUBRUTINA NUMĂRARE BENEFICIARI ÎNSCRIŞI ÎN FIŞIER
1550 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1560 I3=0
1570 FOR I=1 TO 30
1580 IF B$(I)=" " THEN 1600
1590 I3=I3+1
1600 NEXT I
```

```
1610 CLOSE 1
1620 RETURN
1630 STOP
1640 END
```

☐ **Să se proiecteze şi să se realizeze un program** BASIC-PLUS ghidat de un meniu pentru crearea şi întreţinerea fişierului masiv virtual JUCĂRII . DAT avînd structura modificată în: cod articol, denumire articol, a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3, preţ.

☐ **Să se proiecteze şi să se realizeze un program** BASIC-PLUS ghidat de un meniu pentru crearea şi întreţinerea fişierului masiv virtual CONSUMURI . DAT avînd structura modificată în: cod articol, denumire articol, cod provenienţă (R.S.R./IMPORT), consum normat, consum realizat, preţ de achiziţie.

Indicaţie. Se vor utiliza subrutinele din EXEMPLUL 10-M.

CONVERSAȚIA 11

Proiectarea și realizarea unui program BASIC pentru sortarea unui fișier secvențial și a unui fișier masiv virtual de livrări prin metodele: extracție, inserție, SHELL, indexată. Funcții standard pe șiruri de caractere. Instrucțiunea ON ... GOSUB. Funcția SWAP. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLELE 11-PC, m, M, F

☐ De la problemă la program

Vom relua problema din conversaţiile 7, 8 şi 10 complicînd-o sub mai multe aspecte.

Astfel, în EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F vom trata problema creerii şi întreţinerii fişierului secvenţial de livrări cu structură modificată: denumire-beneficiar, cantitate-livrată, preţ. Fişierul va fi creat în cadrul programului principal, iar setul de date privind livrările de benzină va fi supus, pe lîngă operaţiile de adăugare (beneficiari), listare (beneficiari), consultare (beneficiari), salvare (fişier), încărcare (fişier) şi operaţiei de **sortare.**

Sortarea este o operaţie foarte des întîlnită în cadrul programelor de gestiune economică, deoarece majoritatea rapoartelor presupun ca anumite informaţii să fie într-un anume fel aranjate, ceea ce înseamnă ca valorile diferitelor cîmpuri (numite chei) ale unui articol să fi fost deja ordonate, crescător sau descrescător.

În cadrul acestor programe (EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F) referirea la un anume beneficiar se va face nu prin cod ci prin numele beneficiarului.

EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F, puţin mai complicate decît precedentele, vor fi prevăzute cu un meniu care să afişeze pe ecran funcţiunile programului:

1) Adăugare beneficiari
2) Listare beneficiari
3) Consultare beneficiari
4) Sortare date
5) Salvare fişier beneficiari
6) Încărcare fişier beneficiari.

Sortarea (opţiunea 4), ghidată şi ea de un meniu, va fi realizată prin una din metodele: BUBBLE SORT, extracţie, inserţie, sortare indexată, SHELL, QUICK SORT.

Pentru fiecare din opţiunile programului principal şi al subprogramelor de sortare se va scrie cîte o subrutină BASIC ce va fi prevăzută cu mesaje corespunzătoare.

Utilizatorul va tasta o cifră de la 1 la 6 pentru meniul principal şi o cifră de la 1 la 8 pentru submeniul de sortare fără a mai fi obligat să acţioneze tasta **[CR]**. Pentru oprirea execuţiei programului se va tasta cifra zero. Cît priveşte reîntoarcerea în meniul principal şi în submeniul subrutinei de sortare, aceasta se va realiza prin acţionarea tastei **[CR].**

EXEMPLUL 11-M implementat pe minicalculatoarele INDEPENDENT şi CORAL va fi în întregime consacrat sortării fişierului masiv virtual BENEF . DAT creat cu EXEMPLUL 10-M (v. Conversaţia 10).

Programul, pilotat de un meniu, foloseşte subrutinele de sortare din EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F adaptate însă pentru limbajul BASIC-PLUS.

Listele cu subrutinele de sortare vi se vor prezenta în cadrul specificaţiilor de programare urmînd ca dvs. singuri, fără ajutorul nostru, să înţelegeţi mecanismul de proiectare şi realizare a acestora.

Rămîne să vă concentraţi atenţia în special asupra proiectării şi realizării programelor principale ale aplicaţiei: EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F şi EXEMPLUL 11-M.

Vă invităm să testaţi fiecare subrutină de sortare în parte (cu **DATA** şi **READ**) înainte de a o include în programul principal. De reţinut că două (!) din cele opt subrutine scrise pentru EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F conţin cîte o . . . mică eroare de logică. Pentru depanarea acestora vă recomandăm să consultaţi subrutinele scrise (fără erori) în BASIC-PLUS pentru EXEMPLUL 11-M. După cum remarcaţi, de această dată am fost mai puţin generoşi! Vă rugăm să nu ne-o luaţi în nume de rău.

□ Specificaţii de programare

Schema de sistem, structura înregistrării fişierului BENEF . DAT, tabela de variabile, specificaţiile de programare şi sursele subrutinelor (BASIC-80) de sortare sînt prezentate în modulul de analiză structurată, figura 11.1.

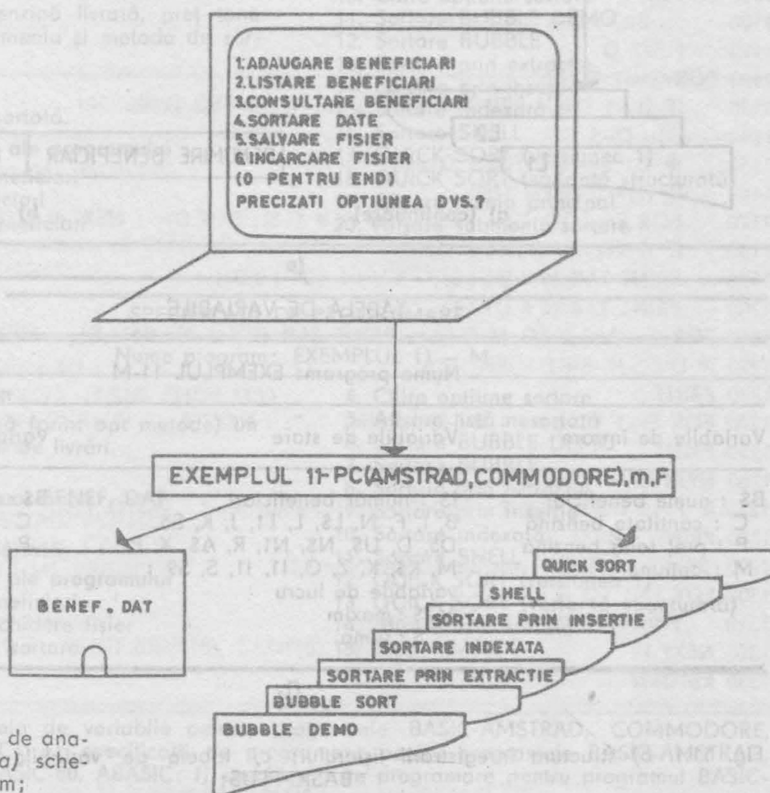
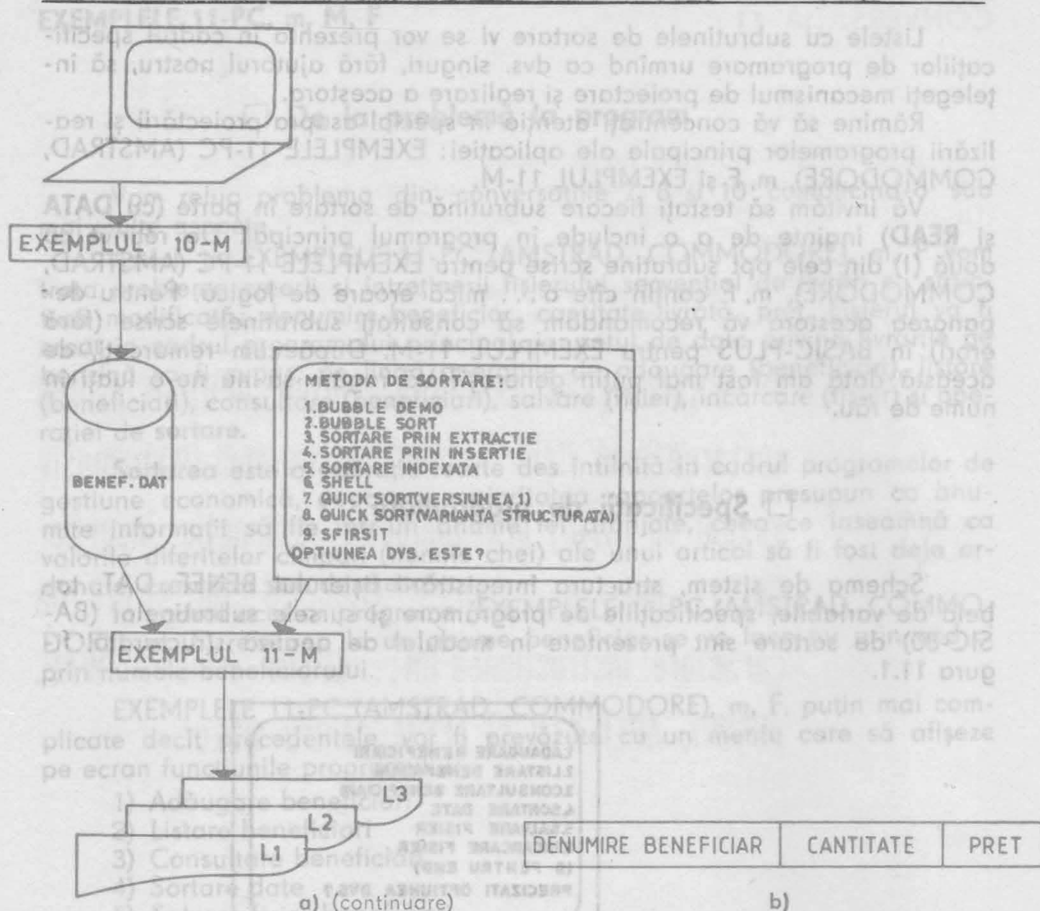


Fig. 11.1. Modulul de analiză structurată: *a)* schema de sistem;



a) (continuare) b)

TABELA DE VARIABILE

Nume program: EXEMPLUL 11-M

Variabile de intrare	Variabile de stare	Variabile de ieșire
B$: nume beneficiar C : cantitate benzină P : preț tonă benzină M : opțiune	I3 : număr beneficiari B, I, F, N, L$, L, L1, J, K, S5, D5, D, U$, N$, N1, R, A$, X, P, M, K$, K, Z, G, I1, J1, S, S9 : variabile de lucru M7$: maxim S : suma	B$: nume beneficiar C : cantitate P : preț

c)

Fig. 11.1. *b)* structura înregistrării fișierului; *c)* tabela de variabile pentru programul BASIC-PLUS;

TABELA DE VARIABILE

Nume program : EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F

Variabile de intrare	Variabile de stare	Variabile de ieşire
BEN$: nume beneficiar CANT : cantitate benzină PREŢ : preţ tonă benzină NBEN$: nume beneficiar căutat OPT : opţiune MET : metoda de sortare A$: continuare/oprire execuţie program	III, I, N, B, F, C, D, D1, E, M, MIN$, J, K STÎNGA, DREAPTA, U$, N1$, R1, R2, K$, N$, S, P, X, G, C1, S1, I1, J1, S9 : variabile de lucru	BEN$: nume beneficiar CANT : cantitate PREŢ : preţ

d)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELELE 11 PC (AMSTRAD, COMMODORE), m, F

Descriere program

Programul creează, întreţine şi sortează un fişier de livrări de organizare secvenţială.

Intrări

Se citesc de la tastatură: nume beneficiar, cantitate benzină livrată, preţ tonă benzină, opţiune meniu şi metoda de sortare.

Ieşiri

Listă beneficiari sortată.

Lista de funcţiuni ale programului

1. Adăugare beneficiari
2. Listare beneficiari
3. Consultare beneficiari
4. Sortare date
5. Salvare fişier beneficiari
6. Încărcare fişier beneficiari
7. Stop
8. Ştergere ecran
9. Citire opţiune meniu
10. Citire opţiune sortare
11. Sortare BUBBLE DEMO
12. Sortare BUBBLE
13. Sortare prin extracţie
14. Sortare prin inserţie
15. Sortare indexată
16. Sortare SHELL
17. QUICK SORT (versiunea 1)
18. QUICK SORT (variantă structurată)
19. Afişare meniu principal
20. Afişare submeniu sortare

e)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLUL 11 – M

Descriere program

Programul sortează (prin opt metode) un fişier masiv virtual de livrări.

Intrări

Fişierul masiv virtual BENEF. DAT

Ieşiri

Listă beneficiari sortată

Lista de funcţiuni ale programului

1. Numărare beneficiari
2. Deschidere-închidere fişier
3. Afişare meniu sortare
4. Citire opţiune sortare
5. Afişare listă nesortată
6. Sortare BUBBLE DEMO
7. Sortare BUBBLE
8. Sortare prin extracţie
9. Sortare prin inserţie
10. Sortare indexată
11. Sortare SHELL
12. QUICK SORT (versiunea 1)
13. QUICK SORT (variantă structurată)
14. Afişare listă sortată
15. Stop

f)

Fig. 11.1. *d)* tabela de variabile pentru programele BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC; *e)* specificaţii de programare pentru programele BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC; *f)* specificaţii de programare pentru programul BASIC-PLUS;

Subrutina BUBBLE DEMO (890)

```
890 REM "BUBBLE DEMO"
900 N=III
910 PRINT "Lista beneficiari (nesortată)"
920 FOR I=1 TO N
930    PRINT BEN$(I), CANT(I), PREŢ(I)
940 NEXT I
950 PRINT : PRINT
960 FOR B=1 TO N-1
970    PRINT "Pasul -->"; B
980    F=0
990    FOR C=1 TO N-B
1000      FOR D=1 TO N
1010         B$(C,D)=BEN$(D)
1020      NEXT D
1030      IF BEN$(C+1)>=BEN$(C) THEN 1080
1040         SWAP BEN$(C), BEN$(C+1)
1050         SWAP CANT(C), CANT(C+1)
1060         SWAP PREŢ(C), PREŢ(C+1)
1070         F=1
1080    NEXT C
1090    FOR D=1 TO N
1100       B$(N-B+1,D)=BEN$(D)
1110    NEXT D
1120 FOR D=1 TO N
1130    IF D=1 THEN PRINT " * "; : D1=-1 : GOTO 1210
1140    D1=D-2
1150    IF D1>=N-B THEN D1=N-B+1
1160    IF D1=0 THEN 1180
1170    FOR E=1 TO D1 : PRINT TAB (E * 6-3); B$(E,D); : NEXT E
1180    IF D>N-B+1 THEN 1220
1190    PRINT TAB (6 * D1+2); "*"; B$ (D1+2,D); :
1200    PRINT TAB (6 * D1+8); "*";
1210 FOR E=D1+2 TO N-B+1 : PRINT TAB (E * 6-3); B$(E,D); : NEXT E
1220 IF D<>N-B+1 THEN 1250
1230 PRINT
1240 FOR E=1 TO 6 * (N-B+1) : PRINT "-"; : NEXT E
1250 PRINT
1260 NEXT D
1270 IF F=0 THEN PRINT : PRINT "==Terminat==" : GOTO 1290
1280 PRINT : NEXT B
1290. PRINT : PRINT "Lista beneficiari (sortată)"
1300 FOR I=1 TO N
1310    PRINT BEN$(I), CANT(I), PREŢ(I)
1320 NEXT I
1330 RETURN
```

g)

Fig. 11.1. *g)* subrutina 890;

Subrutina BUBBLE SORT (1550)

```
1550 REM "Sortare BUBBLE SORT"
1560 I=III
1570 REM
1580 FOR M=1 TO I-1
1590   F=0
1600 FOR N=1 TO I-M
1610   IF BEN$(N+1)>=BEN$(N) THEN 1660
1620     SWAP BEN$(N), BEN$(N+1)
1630     SWAP CANT(N), CANT(N+1)
1640     SWAP PREȚ(N), PREȚ(N+1)
1650     F=1
1660 NEXT N
1670   IF F=0 THEN 1690
1680 NEXT M
1690 PRINT : PRINT "Fișier beneficiari (sortat)"
1700 FOR B=1 TO I : PRINT BEN$(B), CANT(B), PREȚ(B) : NEXT B
1710 PRINT
1720 RETURN
```

h)

Subrutina de sortare prin extracție (1730)

```
1730 REM "Sortare prin extracție"
1740 I=III
1750 FOR M=1 TO I-1
1760   REM "Se consideră că BEN$(M) este minimul"
1770   MIN$=BEN$(M)
1780   J=M
1790   REM
1800   REM
1810   FOR K=M+1 TO I
1820     IF BEN$(K)>=MIN$ THEN 1850
1830       MIN$=BEN$(K)
1840       J=M
1850   NEXT K
1860   REM
1870   SWAP BEN$(M), BEN$(J)
1880   SWAP CANT(M), CANT(J)
1890   SWAP PREȚ(M), PREȚ(J)
1900 NEXT M
1910 PRINT "Lista sortată : "
1920 FOR B=1 TO I : PRINT BEN$(B), CANT(B), PREȚ(B) : NEXT B
1930 RETURN
```

i)

Fig. 11.1. *h)* subrutina 1 550; *i)* subrutina 1 730;

Subrutina de sortare prin inserţie (1940)

```
1940 REM "Sortare prin inserţie"
1950 STÎNGA=1 : DREAPTĂ=III
1960 I=III
1970 FOR M=STÎNGA+1 TO DREAPTĂ
1980   REM "Inserţie"
1990   GOSUB 2050
2000 NEXT M
2010 PRINT "Lista sortată : "
2020 FOR B=1 TO I : PRINT BEN$(B), CANT(B), PREŢ(B) : NEXT B
2030 RETURN
2040 REM "Procedura de inserţie"
2050 PRINT
2060 D=M-1
2070 U$=BEN$(M)
2080 C=CANT(M)
2090 P=PREŢ(M)
2100 IF D<STÎNGA OR U$>=BEN$(D) THEN 2160
2110 BEN$(D+1)=BEN$(D)
2120 CANT(D+1)=CANT(D) : PREŢ(D+1)=PREŢ(D)
2130 D=D-1
2140 GOTO 2100
2150 CANT(D+1)=C : PREŢ(D+1)=P
2160 BEN$(D+1)=U$
2170 RETURN
```

j)

Subrutina SHELL

```
2520 REM "Sortare prin metoda SHELL"
2530 I=III : C=0 : S=0
2540 PRINT "Lista nesortată"
2550 GOSUB 2620
2560 REM "Sortare lista"
2570 GOSUB 2690
2580 PRINT "Lista sortată"
2590 GOSUB 2620
2600 PRINT C; "Comparaţii" : PRINT S; "Schimbări"
2610 RETURN
2620 REM "Tipărire listă nesortată"
2630 FOR K=1 TO I
2640   PRINT BEN$(K); TAB(10); CANT(K); TAB(20); PREŢ(K)
2670 NEXT K
2680 RETURN
2690 REM "Metoda de sortare SHELL"
2700 G=I
2710 WHILE G>1
2720 G=INT (G/2)
2730 M=I-G
2740 F=0
2750 FOR K=1 TO M
2760   P=K+G
2770   C=C+1
2780   IF BEN$(K)<=BEN$(P) THEN 2830
2790   S=S+1
2800   SWAP BEN$(K), BEN$(P)
2810   SWAP CANT(K), CANT(P) : SWAP PREŢ(K), PREŢ(P)
2820   F=1
2830 NEXT K
2840 IF F>0 THEN 2740
2850 WEND
2860 RETURN
```

k)

Fig. 11.1. *j)* subrutina 1 940; *k)* subrutina 2 520;

Subrutina QUICK SORT (versiunea 1)

```
2870 REM "QUICK SORT (versiunea 1)"
2880 N=III : PRINT "Listă nesortată"
2890 I=N : GOSUB 2620
2900 C1=0 : S1=0
2910 I1=1 : J1=N
2920 I=I1 : J=J1 : S=-1
2930 C1=C1+1
2940 IF BEN$(I)<=BEN$(J) THEN 2980
2950 S1=S1+1
2960 SWAP BEN$(I), BEN$(J) : SWAP CANT(I), CANT(J) : SWAP PREȚ(I), PREȚ(J)
2970 S=SGN(-S)
2980 IF S=1 THEN I=I+1 ELSE J=J-1
2990 IF I<J THEN 2930
3000 IF I+1>=J1 THEN 3020
3010 P=P+1 : S9(P,1)=I+1 : S9(P,2)=J1
3020 J1=I-1
3030 IF I1<J1 THEN 2920
3040 IF P=0 THEN 3070
3050 I1=S9(P,1) : J1=S9(P,2) : P=P-1
3060 GOTO 2920
3070 PRINT : PRINT "Lista sortată"
3080 I=N : GOSUB 2620
3090 PRINT : PRINT N; "beneficiari"
3100 PRINT C1; "comparații"
3110 PRINT S1; "schimbări"
3120 PRINT
3130 RETURN
```

l)

Subrutina QUICK SORT (varianta structurată)

```
3200 REM "QUICK SORT (varianta structurată)"
3210 PRINT "Listă nesortată"
3215 I=III : GOSUB 2620
3220 N=III
3230 P=1 : S9(1, 1)=1 : S9(1, 2)=N
3240 WHILE P>0
3250   I1=S9(P, 1) : J1=S9(P, 2) : P=P-1
3260   WHILE I1<J1
3270      I=I1 : J=J1 : S%=-1
3280      WHILE I<J
3290         IF BEN$(I)<BEN$(J) THEN 3340
3300           SWAP BEN$(I), BEN$(J)
3310           SWAP CANT(I), CANT(J)
3320           SWAP PREȚ(I), PREȚ(J)
3330           S%=-(S%+1) :
3340         IF NOT S% THEN J=J-1 ELSE I=I+1
3350      WEND
3360      IF I+1>=J1 THEN 3400
3370         P=P+1
3380         S9(P, 1)=I+1
3390         S9(P, 2)=J1
3400         J1=I-1
3410   WEND
3420 WEND
3430 PRINT
3440 PRINT "Listă sortată"
3450 I=N : GOSUB 2620
3460 RETURN
```

m)

Fig. 11.11. *l)* subrutina 2 870; *m)* subrutina 3 200;

Subrutina de sortare indexată

```
2180 REM "Sortare indexată"
2190 N=III
2200 FOR R=1 TO N
2210   N$(R, 1)=BEN$(R)
2220   R1=CANT(R) : N$(R, 2)=STR$(R1)
2230   R2=PREŢ(R) : N$(R,3)=STR$(R2)
2240 NEXT R
2250 INPUT "Pentru continuare tastaţi (Y/N)' ; A$
2260 WHILE ASC(A$)=89
2270   INPUT "Cîmpul : 1. Beneficiar/2. Cantitate/3. Preţ (1/2/3)"; J
2280   FOR R=1 TO N
2290     K$(R)=N$(R, J)
2300   NEXT R
2310   GOSUB 2430
2320 PRINT "Înregistrări sortate după cîmpul"; J
2330 FOR R=1 TO N
2340   FOR I=1 TO 3
2350       R1=X(R) : PRINT N$(R1, I); " ",
2360   NEXT I
2370   PRINT
2380 NEXT R
2390 PRINT
2400 INPUT "Pentru continuare tastaţi (Y/N)"; A$
2410 WEND
2420 RETURN
2430 FOR A=1 TO N
2440   P=1
2450   FOR B=1 TO N
2460     IF K$(A)>K$(B) THEN P=P+1
2470     IF K$(A)=K$(B) AND A>B THEN P=P+1
2480   NEXT B
2490   X(P)=A
2500 NEXT A
2510 RETURN
```

n)

Fig. 11.1. *n)* subrutina 2 180.

□ Documentaţia de proiectare

Diagrama de structură şi schema logică sînt ilustrate în modulul de proiectare structurată, figura 11.2.

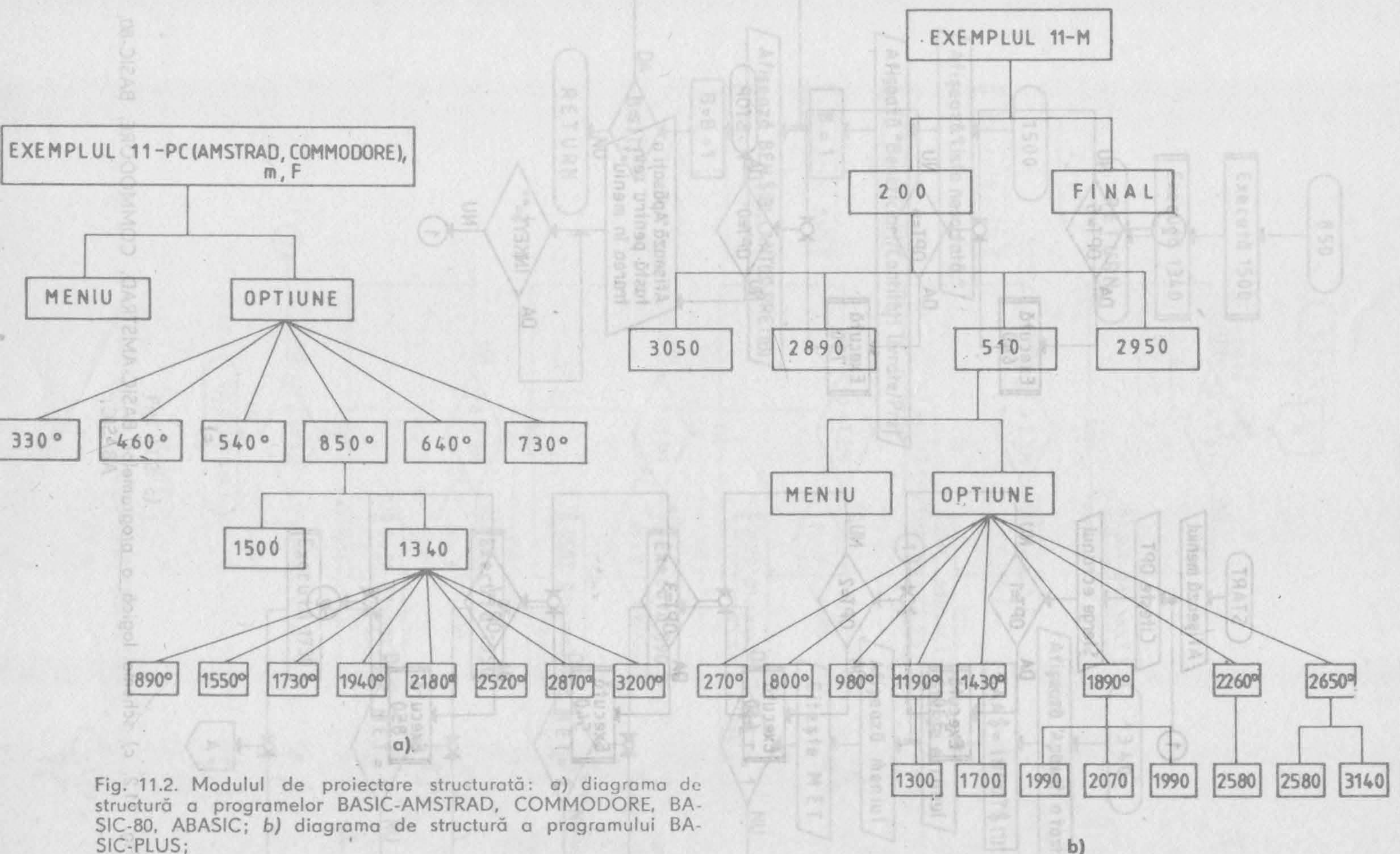


Fig. 11.2. Modulul de proiectare structurată: *a)* diagrama de structură a programelor BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC; *b)* diagrama de structură a programului BASIC-PLUS;

START
Afișează meniul
1
Citește OPT
Șterge ecranul
DA OPT=1 NU
Execută 330
DA OPT=2 NU
Execută 460
DA OPT=3
Execută 540
DA OPT=4
Execută 850
2

2
DA OPT=5 NU
Execută 640
DA OPT=6 NU
Execută 730
NU OPT=0 DA STOP
Afișează "Apăsați o tastă pentru reintrarea în meniu"
DA INKEY$="" NU
1

c)

Fig. 11.2. *c)* schema logică a programelor BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC;

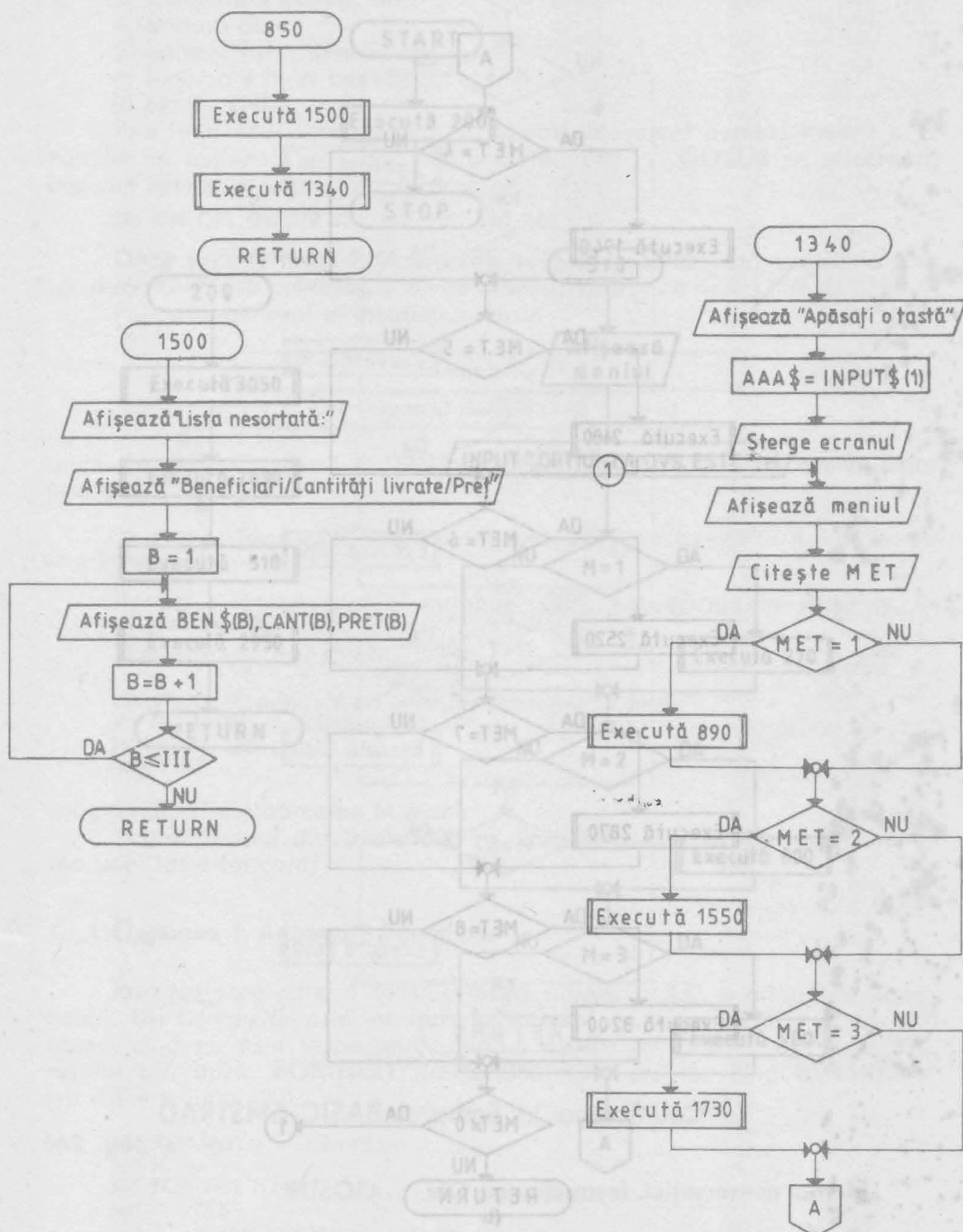

850
Execută 1500
Execută 1340
RETURN
1500
Afişează "Lista nesortată:"
Afişează "Beneficiari/Cantităţi livrate/Preţ"
B = 1
Afişează BEN $(B), CANT(B), PRET(B)
B=B+1
DA
B≤III
NU
RETURN
1340
Afişează "Apăsaţi o tastă"
AAA$= INPUT$(1)
Şterge ecranul
Afişează meniul
Citeşte MET
DA
MET = 1
NU
Execută 890
DA
MET = 2
NU
Execută 1550
DA
MET = 3
NU
Execută 1730
A

Fig. 11.2. c)

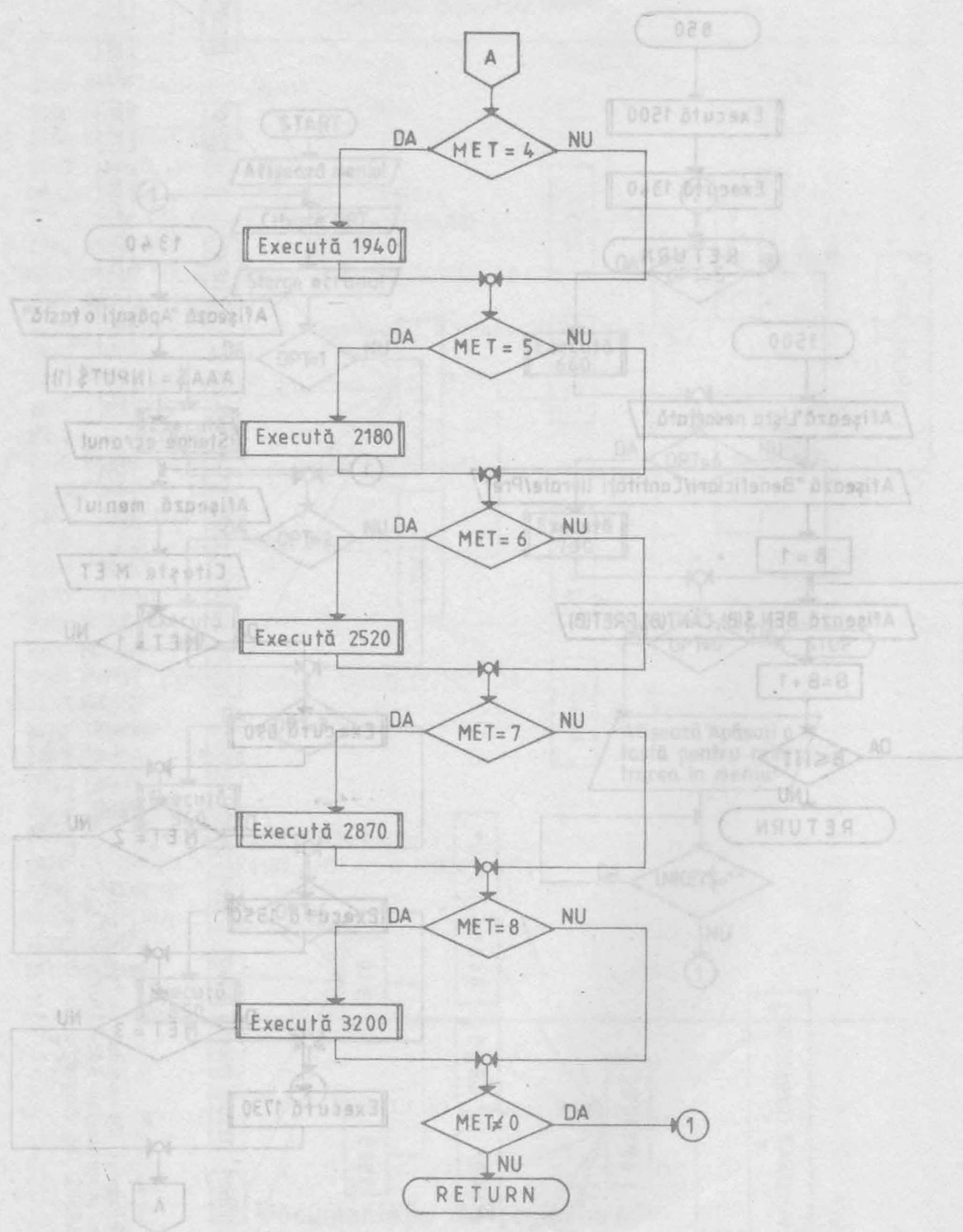
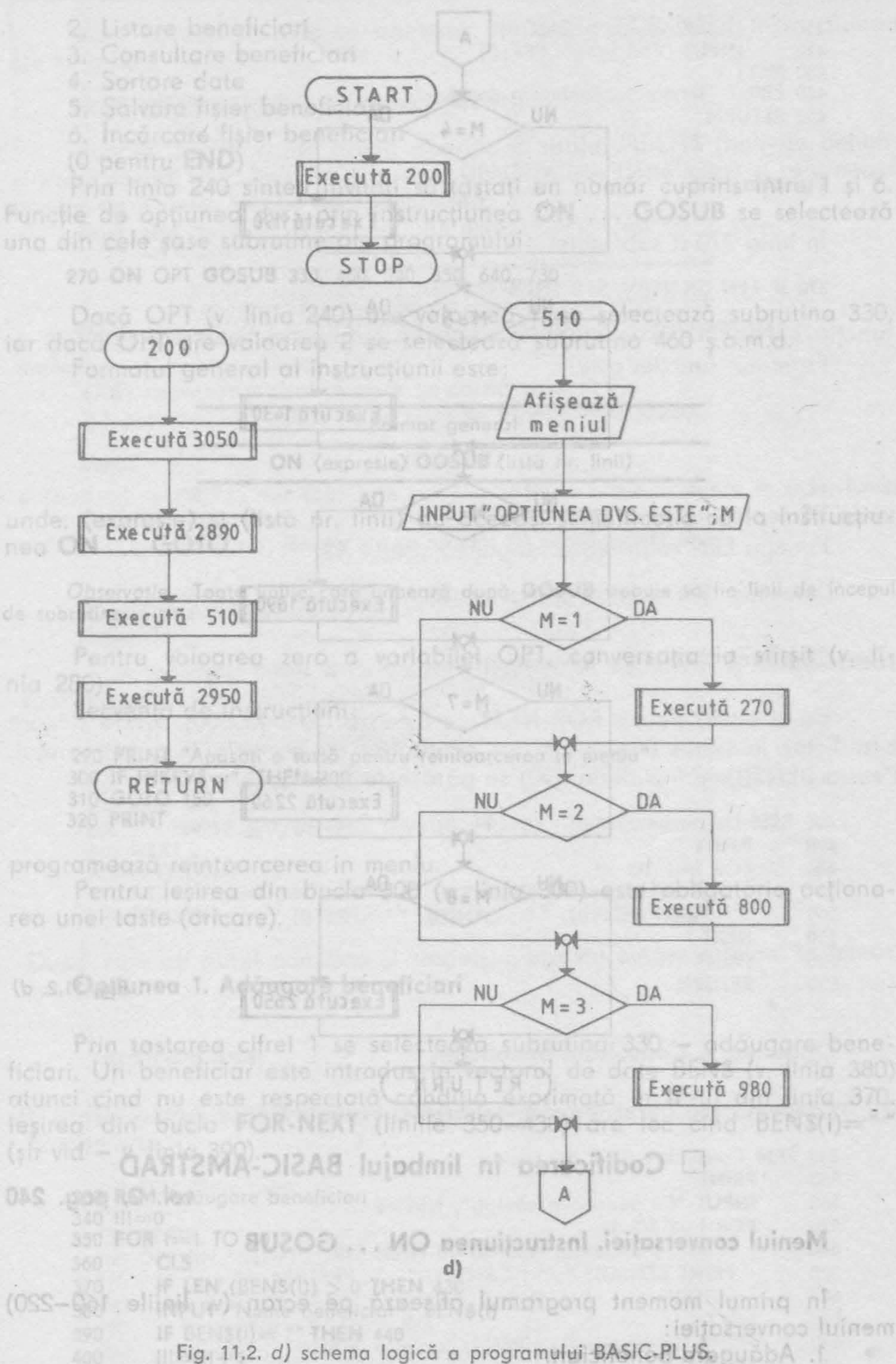

A
DA
MET = 4
NU
Execută 1940
DA
MET = 5
NU
Execută 2180
DA
MET = 6
NU
Execută 2520
DA
MET = 7
NU
Execută 2870
DA
MET = 8
NU
Execută 3200
MET≠0
DA
1
NU
RETURN

Fig. 11.2. c)



d)

Fig. 11.2. *d)* schema logică a programului BASIC-PLUS.

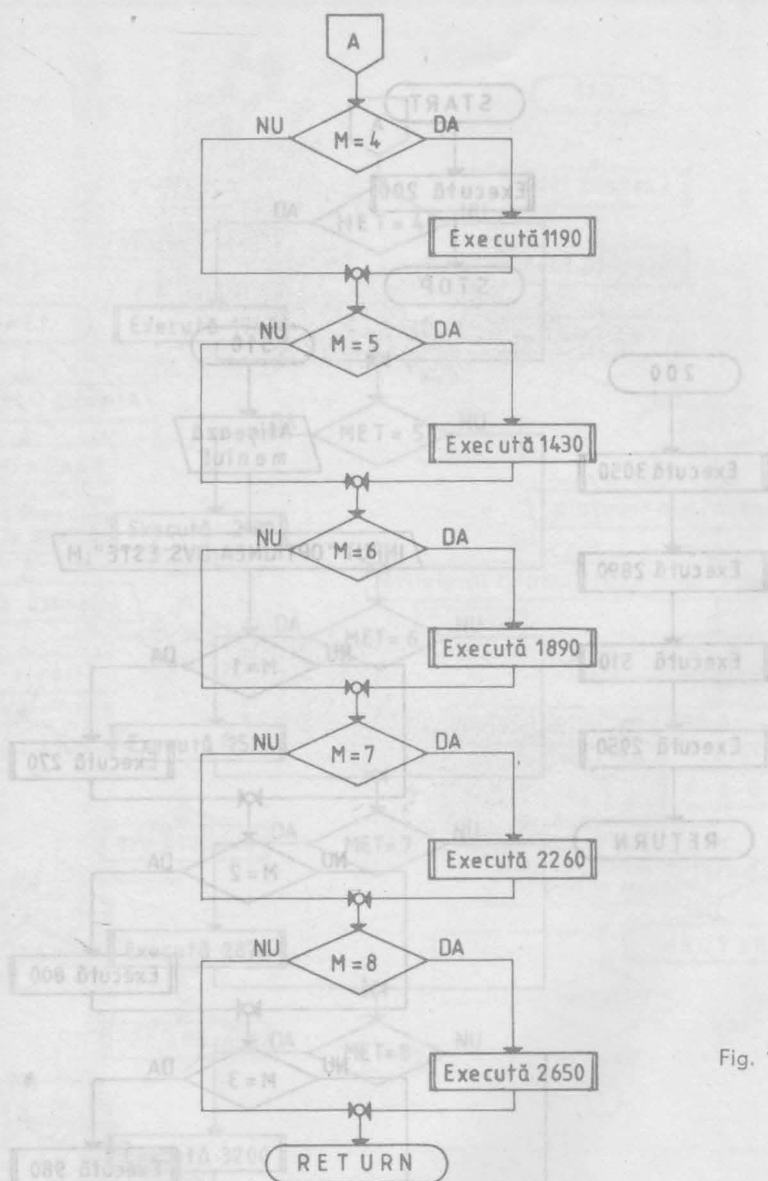


Fig. 11.2. *d)*

☐ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 240

Meniul conversației. Instrucțiunea ON . . . GOSUB

În primul moment programul afişează pe ecran (v. liniile 160–220) meniul conversației:

1. Adăugare beneficiari.

2. Listare beneficiari
3. Consultare beneficiari
4. Sortare date
5. Salvare fişier beneficiari
6. Încărcare fişier beneficiari
(0 pentru **END**)

Prin linia 240 sînteţi invitaţi să tastaţi un număr cuprins între 1 şi 6. Funcţie de opţiunea dvs., prin instrucţiunea **ON** ... **GOSUB** se selectează una din cele şase subrutine ale programului:

```
270 ON OPT GOSUB 330, 460, 540, 850, 640, 730
```

Dacă OPT (v. linia 240) are valoarea 1, se selectează subrutina 330, iar dacă OPT are valoarea 2 se selectează subrutina 460 ş.a.m.d.

Formatul general al instrucţiunii este:

Format general
ON ⟨expresie⟩ **GOSUB** ⟨listă nr. linii⟩

unde, ⟨expresie⟩ şi ⟨listă nr. linii⟩ au aceeaşi semnificaţie ca la instrucţiunea **ON** ... **GOTO**.

Observaţie. Toate liniile care urmează după **GOSUB** trebuie să fie linii de început de subrutine.

Pentru valoarea zero a variabilei OPT, conversaţia ia sfîrşit (v. linia 280).

Secvenţa de instrucţiuni:

```
290 PRINT "Apăsaţi o tastă pentru reîntoarcerea în meniu"
300 IF INKEY$=" "THEN 300
310 GOTO 150
320 PRINT
```

programează reîntoarcerea în meniu.

Pentru ieşirea din bucla 300 (v. linia 300) este obligatorie acţionarea unei taste (oricare).

Opţiunea 1. Adăugare beneficiari

Prin tastarea cifrei 1 se selectează subrutina 330 – adăugare beneficiari. Un beneficiar este introdus în vectorul de date BEN$ (v. linia 380) atunci cînd nu este respectată condiţia exprimată în **IF**-ul din linia 370. Ieşirea din bucla **FOR-NEXT** (liniile 350–430) are loc cînd BEN$(I)=" " (şir vid – v. linia 390).

```
330 REM Adăugare beneficiari
340 III=0
350 FOR I=1 TO 30
360     CLS
370     IF LEN (BEN$(I)) > 0 THEN 430
380     INPUT "Nume beneficiar"; BEN$(I)
390     IF BEN$(I)= "" THEN 440
400     III=III+1
```

```
410      INPUT "Livrări"; CANT(I)
420      INPUT "Preţ tonă"; PREŢ(I)
430 NEXT I
440 PRINT "Terminat introducere date"
450 RETURN
```

Funcţia LEN

În linia 370 a subrutinei

```
370 IF LEN (BEN$(I)) > 0 THEN 430
```

funcţia **LEN**(BEN$(I)) calculează lungimea şirului de caractere BEN$(I).

Formatul funcţiei este:

Format general
LEN ((X$))

unde X$ reprezintă şirul de caractere.

Funcţia **LEN** returnează lungimea şirului X$.

Observaţie. Sînt numărate atît caracterele netipăribile cît şi spaţiile.

Opţiunea 2. Listare beneficiari

Cu o buclă de tip **FOR-NEXT** se listează (v. linia 500) pentru fiecare beneficiar în parte (maximum 30) cantitatea de benzină livrată şi preţul. Pentru BEN$(I)=" " (v. linia 490) se părăseşte forţat bucla.

```
460 REM Listare beneficiari
470      PRINT
480      FOR I=1 TO 30
490        IF BEN$(I)= " " THEN 520
500        PRINT BEN$(I); " " ; CANT(I); " " ; PREŢ(I)
510      NEXT I
520      PRINT "Terminat listare fişier"
530      RETURN
```

Opţiunea 3. Consultare beneficiari

Prin tastarea cifrei 3 se selectează subrutina 540 (v. liniile 540–630).

```
540 REM Consultare fişier beneficiari
550      PRINT
560      INPUT "Ce beneficiar căutaţi"; NBEN$
570      FOR I=1 TO 30
580        IF INSTR (BEN$(I), NBEN$)=0 THEN 610
590        PRINT BEN$(I); "  " ; CANT(I); "  " ; PREŢ(I)
600        RETURN
610      NEXT I
620      PRINT "Beneficiarul"; NBEN$; "nu este în fişier"
630      RETURN
```

Vom decupa în cele ce urmează din bucla **FOR-NEXT** instrucţiunea din linia 580.

```
580 IF INSTR (BEN$(I), NBEN$)=0 THEN 610
```

Funcţia **INSTR** caută prima apariţie a şirului NBEN$ (numele beneficiarului căutat – v. linia 560) în şirul BEN$ (nume beneficiar) şi returnează poziţia de la care începe corespondenţa.

Formatul general al instrucţiunii este:

Format general
INSTR ([⟨i⟩,]⟨X$⟩,⟨Y$⟩)

unde:

⟨X$⟩ reprezintă şirul în care se caută şirul ⟨Y$⟩;
⟨i⟩ precizează poziţia de la care se începe căutarea.

Reguli

- Dacă ⟨i⟩ >**LEN**(X$) sau ⟨X$⟩ este nul, sau ⟨Y$⟩ nu poate fi regăsit în ⟨X$⟩, funcţia returnează valoarea zero.
- Dacă ⟨Y$⟩ este şir nul, funcţia are ca rezultat ⟨i⟩ (dacă există) sau 1.

Opţiunea 5. Salvare fişier

Prin tastarea cifrei 5 (am sărit intenţionat opţiunea 4) se selectează subrutina 640.

```
640 REM Salvare fişier beneficiari
650 PRINT
660 OPENOUT "BENEF. DAT"
670 FOR I=1 TO 30
680     WRITE #9, BEN$(I), CANT(I), PREŢ(I)
690 NEXT I
700 CLOSEOUT
710 PRINT "= =Fişier salvat= ="
720 RETURN
```

După cum aţi putut constata şi singuri, nimic nu este complicat în această subrutină.

Încărcare fişier

Prin tastarea cifrei 6 se selectează subrutina (730) de încărcare fişier în memorie.

```
730 REM Citire fişier
740 III=0
750 PRINT
760 OPENIN "BENEF. DAT"
770 FOR I=1 TO 30
780     INPUT #9, BEN$(I), CANT(I), PREŢ(I)
790     IF BEN$(I)=" " THEN 820
800     III=III+1
810 NEXT I
```

```
820 CLOSEIN
830 PRINT "Fişier încărcat"
840 RETURN
```

De notat că subrutina încarcă şi memorie cel mult 30 de beneficiari. În momentul cînd BEN$(I)="" (şir vid) fişierul BENEF.DAT este închis şi pe ecran apare mesajul „Fişier încărcat".

Opţiunea 4. Sortare date

Iată-ne ajunşi la opţiunea 4. Prin tastarea cifrei 4 se selectează subrutina 850.

```
850 REM Subrutină de sortare date
860 GOSUB 1500
870 GOSUB 1340
880 RETURN
```

Subrutina 1500 se apelează pentru afişarea listei nesortate a beneficiarilor.

```
1500 REM
1510 PRINT "Lista nesortată : " : PRINT
1520 PRINT "Beneficiari/Cantităţi livrate/Preţ" : PRINT
1530 FOR B=1 TO III : PRINT BEN$(B); TAB(21); CANT(B); TAB(31); PREŢ(B) :
     NEXT B
1540 RETURN
```

Prin apelarea subrutinei 1340 se afişează (v. liniile 1360–1450) submeniul subrutinei de sortare.

```
1340 PRINT : PRINT "Apăsaţi o tastă"
1345 IF INKEY$= "" GOTO 1345
1350 CLS
1360 PRINT TAB(5); " Metoda de sortare : "
1370 PRINT TAB(7); " 1. BUBBLE DEMO"
1380 PRINT TAB(7); " 2. BUBBLE SORT"
1390 PRINT TAB(7); " 3. Sortare prin extracţie"
1400 PRINT TAB(7); " 4. Sortare prin inserţie"
1410 PRINT TAB(7); " 5. Sortare indexată"
1420 PRINT TAB(7); " 6. SHELL"
1430 PRINT TAB(7); " 7. QUICK SORT (versiunea 1)"
1440 PRINT TAB(7); " 8. QUICK SORT (varianta structurată)"
1450 PRINT TAB(7); "(0 pentru END)"
1460 PRINT : PRINT TAB(5); INPUT "Metoda (1–8)"; MET
1470 ON MET GOSUB 890, 1550, 1730, 1940, 2180, 2520, 2870, 3200
1480 IF MET < > 0 THEN 1340
1490 RETURN
```

BUBBLE DEMO

Prin tastarea cifrei 1 se selectează subrutina 890 (v. liniile 890–1330) care constituie un program demonstrativ, pe paşi (v. linia 970) privind modul de lucru al sortării prin metoda "bulelor de aer" (BUBBLE, in l. engleză).

Observaţie. Instrucţiunile 1040, 1050 şi 1060 din subrutina 890 listată în fig. 11.1. g se vor inlocui cu:

```
1040 V$=BEN$(C)
1041 BEN$(C)=BEN$(C+1)
1042 BEN$(C+1)=V$
1050 Z=CANT(C)
1051 CANT(C)=CANT(C+1)
1052 CANT(C+1)=Z
1060 U=PREŢ(C)
1061 PREŢ(C)=PREŢ(C+1)
1062 PREŢ(C+1)=U
```

BUBBLE SORT

Prin tastarea cifrei 2 se selectează subrutina 1550 (v. liniile 1550–1720). Subrutina precedentă a fost scrisă pentru a vă ajuta să înţelegeţi cum lucrează sortarea BUBBLE. Dar, într-o sortare "normală" a datelor nu este nevoie să vedeţi fiecare trecere. Subrutina va imprima doar lista finală sortată (v. rezultatele execuţiei). Subrutina compară cheile, două cîte două şi schimbă elementele dacă nu este îndeplinită condiţia din linia 1610.

Remarcă. Instrucţiunile din liniile 1620, 1630, 1640 se vor înlocui în aceeaşi manieră în care am procedat cu subrutina 890 (v. observaţie).

Sortare prin extracţie

Prin tastarea cifrei 3 se selectează subrutina 1730 (v. liniile 1730–1930). Subrutina extrage elementul avînd cheia cu valoarea cea mai mică şi îl schimbă cu elementul din stînga tabloului.

Remarcă. Instrucţiunile BASIC-AMSTRAD din liniile 1870, 1880 şi 1890 se vor înlocui în aceeaşi manieră în care am procedat la subrutina 890 (v. observaţie).

Sortare prin inserţie

Prin tastarea cifrei 4 se selectează subrutina 1940 (v. liniile 1940–2020). Subrutina inserează un element al tabloului într-o zonă deja ordonată a acestui tablou, după care prelucrează elementele care rămîn.

Sortare indexată

Prin tastarea cifrei 5 se selectează subrutina 2250 (v. liniile 2250–2510). Subrutina sortează fişierul de livrări după trei chei (la alegere): Beneficiar, Cantitate şi Pret. Cele trei cîmpuri de sortare se selectează prin tastarea 1/2/3.

Funcţia STR$

În liniile 2220 şi 2230 ale subrutinei

```
2220 R1=CANT(R) : N$(R,2)=STR$(R1)
2230 R2=PREŢZ(R) : N$(R,3)=STR$(R2)
```

funcţia **STR$** converteşte valoarea lui R1 şi R2 într-un şir de caractere. Poate vi se pare curios efectul acestei funcţii, dar este mult mai uşor de manipulat un număr sub formă de şir decît sub forma lui numerică.

Formatul general al funcţiei este:

Format general
STR$ (⟨expresie numerică⟩)

Sortarea SHELL

Prin tastarea cifrei 6 se selectează subrutina 2520. Subrutina SHELL, numită astfel după numele inventatorului metodei de sortare – SHELL, este mult mai rapidă decît cele prezentate anterior.

SHELL a avut ideea efectuării unor salturi importante în tablou pentru plasarea mai rapidă a elementelor. Schimbarea pe o "distanţă" mai mare a elementelor este mult mai eficientă decît efectuarea metodică de schimbări ale unei singure poziţii deodată. "Distanţa" dintre elementele comparate se numeşte *interval* iar în linia 2700 este readusă la G=I, dar schimbată în G/2 în linia 2720.

Deci, pentru I=30 elemente, intervalul de pornire este **INT** (30/2)=15. În cele din urmă, G este adus la un "mic interval" egal cu 1. Cu alte cuvinte, subrutina este identică cu BUBBLE. Dar, de data aceasta datele vor fi "aproape ordonate". Variabila F face ca programul să se oprească atunci cînd nu se mai pot face schimbări (F=0). Pentru a înţelege bine subrutina va trebui să consideraţi cel puţin 16 valori ale tabloului, folosind un interval de comparaţie de pornire – 8.

Remarcă. Instrucţiunile **SWAP** din liniile 2800 şi 2810 se vor înlocui în aceeaşi manieră în care am procedat cu subrutina 890 (v. observaţie).

Subrutina afişează numărul de comparaţii (C) şi numărul de schimbări (S) efectuate.

QUICK SORT (versiunea 1)

Prin tastarea cifrei 7 se selectează subrutina 2870 (v. liniile 2870–3130).

Metoda QUICK SORT (sortare rapidă) este foarte mult apreciată de către profesionişti (este de ordinul n * log n). Metoda este foarte asemănătoare cu modul în care unii oameni efectuează sortarea natural (împărţirea unei probleme în subprobleme ş.a.m.d).

Funcţia aritmetică SGN. Funcţia **SGN** din linia 2970 stabileşte semnul lui S.

```
2970 S=SGN(–S)
```

SGN returnează valorile: –1 (dacă valoarea este negativă), 0 (dacă ea este nulă) şi 1 (dacă ea este pozitivă).

Formatul funcţiei este:

Format general
SGN (⟨expresie numerică⟩)

Remarcă. Instrucţiunile **SWAP** din linia 2960 se vor înlocui în aceeaşi manieră în care am procedat cu subrutina 890 (v. observaţie).

QUICK SORT (varianta structurată)

Prin tastarea cifrei 8 se selectează subrutina 3200. Ea reprezintă varianta structurată a subrutinei QUICK SORT anterioare (2870). De remarcat prezenţa variabilelor S9, S%. Atenţie la liniile 3330 şi 3340!

Remarcă. Instrucţiunile **SWAP** din liniile 3300 şi 3320 se vor înlocui în aceeaşi manieră în care am procedat cu subrutina 890 (v. observaţie).

TEST

Următorul program nestructurat, scris în BASIC HC-85, TIM S, SPECTRUM realizează căutarea într-un şir, prin metoda bisecţiei, a unui anumit nume. Extindeţi metoda pentru un fişier BASIC–AMSTRAD secvenţial sortat după o anumită cheie.

```
 10 CLEAR
 20 DIM n$ (20, 6)
 30 DIM t$ (20, 4)
 40 LET i=1
 50 INPUT n$(I)
 60 INPUT t$(I)
 65 IF n$(i)=" * " THEN GO TO 90
 70 LET i=i+1
 80 GO TO 50
 90 REM
100 LET n=i
150 INPUT "Ce nume căutaţi"; q$
210 PRINT "1"; TAB 5; "h"; TAB 10;
    "m"; TAB 15; "n$(m)"
215 REM l → limita inferioară;
    h → limita superioară
216 REM m → media
220 LET l=1
225 LET h=n
230 REM
235 IF h–l=1 THEN GOTO 500
240 LET m=INT ((l+h)/2)
250 PRINT l; TAB 5; h; TAB 10; m;
    TAB 15; n$(m)
260 IF q$=n$(m) THEN GOTO 320
270 IF q$<n$(m) THEN GOTO 300
280 LET l=m
290 GOTO 230
300 LET h=m
310 GOTO 230
320 REM
330 PRINT q$; t$(m)
350 GOTO 600
510 PRINT q$; "nu este în listă"
600 INPUT "Doriţi să căutaţi alt, nume";
    Y$
610 IF Y$="DA" THEN GOTO 150
620 STOP
```

□ Codificarea în limbajul BASIC-COMMODORE

În comparaţie cu programul BASIC–AMSTRAD, diferenţele de scriere a programului apar în: linia 260, 300, 360, 580 (limbajul BASIC-COMMODORE nu acceptă funcţia **INSTR**), 660, 680, 700, 760, 820, 1040, 1050, 1060, (limbajul BASIC-COMMODORE nu acceptă funcţia **SWAP**), 1350, 1620, 1630, 1640, 1870, 1880, 1890, 2260, 2410, 2710, 2800, 2810, 2850, 2960, 3240, 3260, 3280, 3300, 3310, 3320, 3350, 3410, 3420.

Aplicație. Se consideră un fișier dicționar (cod, denumire) secvențial cu 30 de articole. Fișierul este sortat ascendent după cod. Apelind una din subrutinele de mai jos (verifică dacă un număr K se află în vectorul A cu 30 elemente), să se cerceteze dacă un anume beneficiar al cărui cod se citește în variabila A se găsește în fișierul dicționar.

a)

```
10 DIM A(30)
70 I=1
75 IF K=A(I) THEN 120
80 I=I+1
90 IF I<=30 THEN 75
100 PRINT "NUMĂRUL"; K; "NU
    ESTE ÎN VECTOR"
110 END
120 PRINT I, A(I)
130 RETURN
```

b)

```
60 Q=INT (SQR(30))
70 I=1-Q
80 INPUT "NR. CĂUTAT"; K
90 I=I+Q
100 IF I>=30 THEN 190
110 IF K>A(I) THEN 90
120 IF K=A(I) THEN 240
130 IF I=1 THEN 170
140 FOR J=I-1 TO I-Q+1 STEP -1
150     IF K=A(J) THEN 230
160 NEXT J
170 PRINT
180 END
190 FOR J=30 TO I-Q+1 STEP -1
200     IF K=A(J) THEN 230
210 NEXT J
220 GOTO 170
230 I=J
240 PRINT I, A(I)
250 PRINT
260 RETURN
```

c)

```
70 L=1
80 U=30
90 IF U<L THEN 200
100 I=INT (L+U)/2
110 IF K<A(I) THEN 180
120 IF K=A(I) THEN 150
130 L=I+1
140 GO TO 90
150 PRINT I, A(I)
160 PRINT
170 END
180 U=I-1
190 GO TO 90
200 PRINT
210 RETURN
```

□ Particularități ale programării în limbajul BASIC-80

vol. 2, pag. 248

Funcția SWAP

Lista programului o puteți consulta în vol. 2, pag. 248. În cele ce urmează ne vom opri asupra unui singur element de . . . noutate, comparativ cu programele precedente. Este vorba de funcția **SWAP** pe care ați întîlnit-o în toate subrutinele de sortare, mai puțin subrutina 1730 (sortare prin extracție).

```
1040 SWAP BEN$(C), BEN$(C+1)
1050 SWAP CANT(C), CANT(C+1)
1060 SWAP PREȚ(C), PREȚ(C+1)
```

SWAP schimbă între ele valorile a două variabile. În linia 1040 valoarea lui BEN$(C) trece în locul valorii lui BEN$(C+1) și invers.

Formatul funcției **SWAP** este:

Format general
SWAP ⟨variabilă⟩, ⟨variabilă⟩

Remarcă. Cele două variabile trebuie să aibă același tip (întreg, simplă precizie, dublă precizie, șir).

Aplicaţie. Introduceţi, executaţi şi analizaţi următoarele programe:

a)
```
10 REM
20 INPUT "Introduceţi un cuvînt", a$
30 i=1
40 na=0
50 cont=1
60 WHILE cont=1
70    poz=INSTR (i, a$, "a")
80    IF poz>0 THEN na=na+1 : i=i+1 ELSE cont=0
90 WEND
100 PRINT "Litera a apare de"; na; "in cuvintul dvs."
```

b)
```
20 INPUT "Introduceţi data curentă zz/ll/aa"; data$
30 zi=VAL (data$)
40 mp=INSTR (data$, "/")
50 mh$=MID$ (data$, mp+1)
60 mh=VAL (mh$)
70 ypoz=INSTR (mp+1, data$, "/")
80 yp$=MID$ (data$, ypoz+1)
90 FOR i=1 TO mh
100    READ a$, b
110 NEXT
120 zi=zi+7
130 IF zi>b THEN zi=zi-b : READ a$, b
140 wf$=a$+STR$(zi)+", 19"+yp$
150 PRINT : PRINT wf$
160 DATA Ianuarie, 31, Februarie, 28, Martie, 31, Aprilie, 30, Mai, 31, Iunie, 30
170 DATA Iulie, 31, August, 31, Septembrie, 30, Octombrie, 31, Noiembrie, 30,
    Decembrie, 31
```

Observaţie. Funcţia **MID$** returnează un subşir de ⟨j⟩ caractere începînd cu poziţia ⟨i⟩.

Formatul general al funcţiei este:

Format general
**MID$** (⟨x$⟩, [i, j])

c)
```
10 A$="12345ABC"
20 NR=VAL(A$)
30 PRINT NR
40 END
```

d)
```
10 REM
20 FOR i=65 TO 90
30    PRINT CHR$(i)
40 NEXT
```

Observaţie. Funcţia **CHR** (i) generează un caracter cu codul ASCII ⟨i⟩.

e)
```
10 REM
20 FOR i=0 TO 31
80    PRINT CHR$(1); CHR$(i);
40 NEXT i
```

f)
```
10 REM
20 i=1
30 WHILE i=1
40    a$=INKEY$
50    IF a$ < > " " THEN a=ASC(a$)
60    IF a>64 AND a<91 THEN
      PRINT a$
70 WEND
```

g)
```
10 PRINT
20 INPUT "Introduceţi codul"; cod
30 PRINT
40 LINE INPUT "Introduceţi mesaj";
   mesaj$
50 lung=LEN (mesaj$)
60 codmesaj$=" "
70 FOR i=1 TO lung
80 litera$=MID$ (mesaj$, i, 1)
90 literaascii=ASC (litera$)
100 codliteră$=CHR$ (literaascii+cod)
```

```
110 codmesaj$=codmesaj$+
    codliteră$
120 NEXT
130 PRINT
140 PRINT "Codul mesajului este:"
150 PRINT
160 PRINT codmesaj$
170 END
```

Observație. Instrucțiunea **LINE INPUT** (v. linia 40) citește o linie terminată cu [**CR**] în variabila de tip șir "mesaj$" de la consolă (v. vol. 2, pag. 269).

h)

```
1000 REM "SORTARE DUPĂ TABEL"
1010 DIM BEN$20), R(20, 20)
1020 INPUT "NUMĂR REZERVOARE : ", N : INPUT "NUMĂR BENEFICIARI : ", M
1030 INPUT "NUMĂRUL REZERVORULUI DUPĂ CARE SE FACE SORTAREA", L
1040 FOR I=1 TO M
1050   PRINT "NUMELE BENEFICIARULUI NR.", I, " "; : INPUT BEN$(I)
1060 NEXT I
1070 FOR I=1 TO N
1080   PRINT " *** REZERVORUL"; I
1090   FOR J=1 TO M
1100     PRINT "CANTITATEA LIVRATĂ BENEFICIARULUI";
         BEN$(J); : INPUT R(I, J)
1110   NEXT J
1120 NEXT I
1130 LPRINT "Lista nesortată : "
1140 GOSUB 1270
1150 FOR I=1 TO M
1160   FOR K=I+1 TO M
1170     IF R(L, I)>=R(I, K) THEN 1220
1180     SWAP BEN$(I), BEN$(K)
1190     FOR J=1 TO N
1200         SWAP R(J, I), R(J, K)
1210     NEXT J
1220   NEXT K
1230 NEXT I
1240 LPRINT "Sortare după rezervorul"; L
1250 GOSUB 1270
1260 END
1270 LPRINT "..............................................................."
1280 LPRINT " ! NUME BENEF.    !   REZ1   !   REZ2   !   REZ3   !   REZ4 ! "
1290 LPRINT "..............................................................."
1300 FOR I=1 TO M
1310 LPRINT " ! " ; BEN$(I); TAB (16);
1320 FOR J=1 TO N
1330   LPRINT USING "###.###"; R(J, I);
1340   LPRINT " ! " ;
1350 NEXT J
1360 LPRINT
1370 NEXT I
1380 LPRINT "..............................................................."
1390 RETURN
```

Lista nesortată:

! NUME BENEF. !	REZ.1 !	REZ2 !	REZ3 !	REZ4 !
! PECO BUZĂU	4.678 !	36.911 !	211.600 !	5.947 !
! PECO CLUJ	6.975 !	43.870 !	321.343 !	243.199 !
! COMB. BRAZI	12.457 !	24.766 !	3.203 !	56.978 !

Sortare după rezervorul 4

! NUME BENEF. !	REZ1 !	REZ2 !	REZ3 !	REZ4 !
! PECO CLUJ	6.975 !	43.870 !	321.343 !	243.199 !
! COMB. BRAZI	12.457 !	24.766 !	3.203 !	56.978 !
! PECO BUZĂU	4.678 !	36.911 !	211.600 !	5.947 !

□ Particularităţi ale programării în limbajul BASIC-PLUS

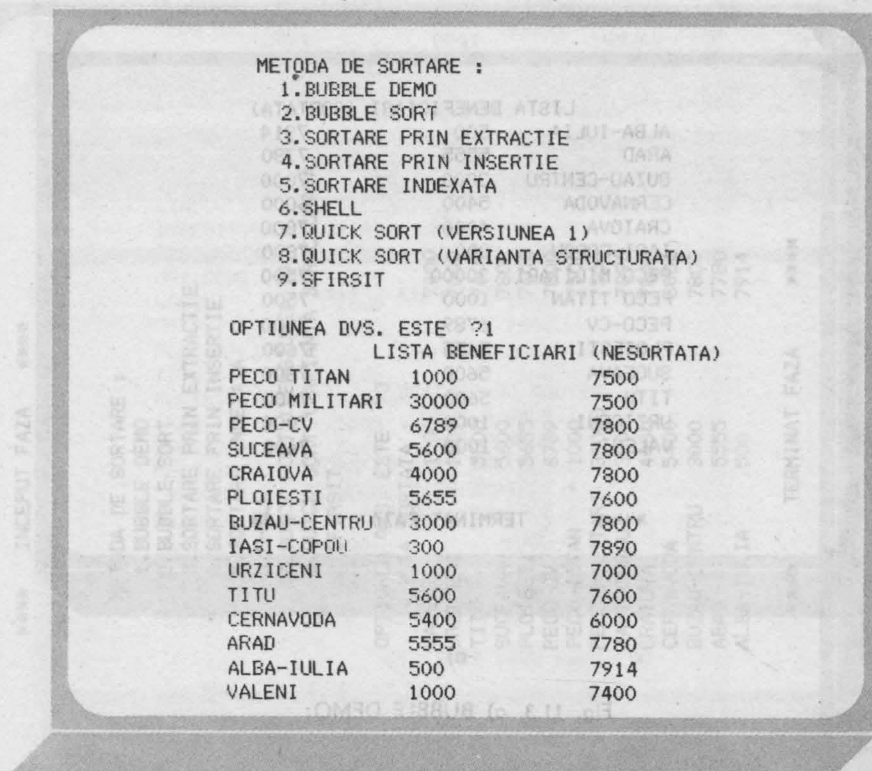
vol. 2, pag. 255

Programul sortează fişierul masiv virtual BENEF . DAT creat în conversaţia 10. Pentru început, este apelată subrutina 220 (v. linia 180). La rîndul ei, subrutina apelează patru subrutine: 3050 (numărare beneficiari); 2890 (deschidere fişier); 510 (corp program+meniu) şi 2950 (închidere fişier).

Conversaţia propriu-zisă începe în subrutina 510 odată cu afişarea meniului. În variabila M se citeşte dinamic opţiunea utilizatorului. Se selectează pe rînd subrutinele: 270 (BUBBLE DEMO), 800 (BUBBLE SORT), 980 (sortare prin extracţie), 1190 (sortare prin inserţie), 1430 (sortare indexată), 1890 (SHELL), 2260 (QUICK SORT – versiunea 1), 2650 (QUICK SORT – varianta structurată).

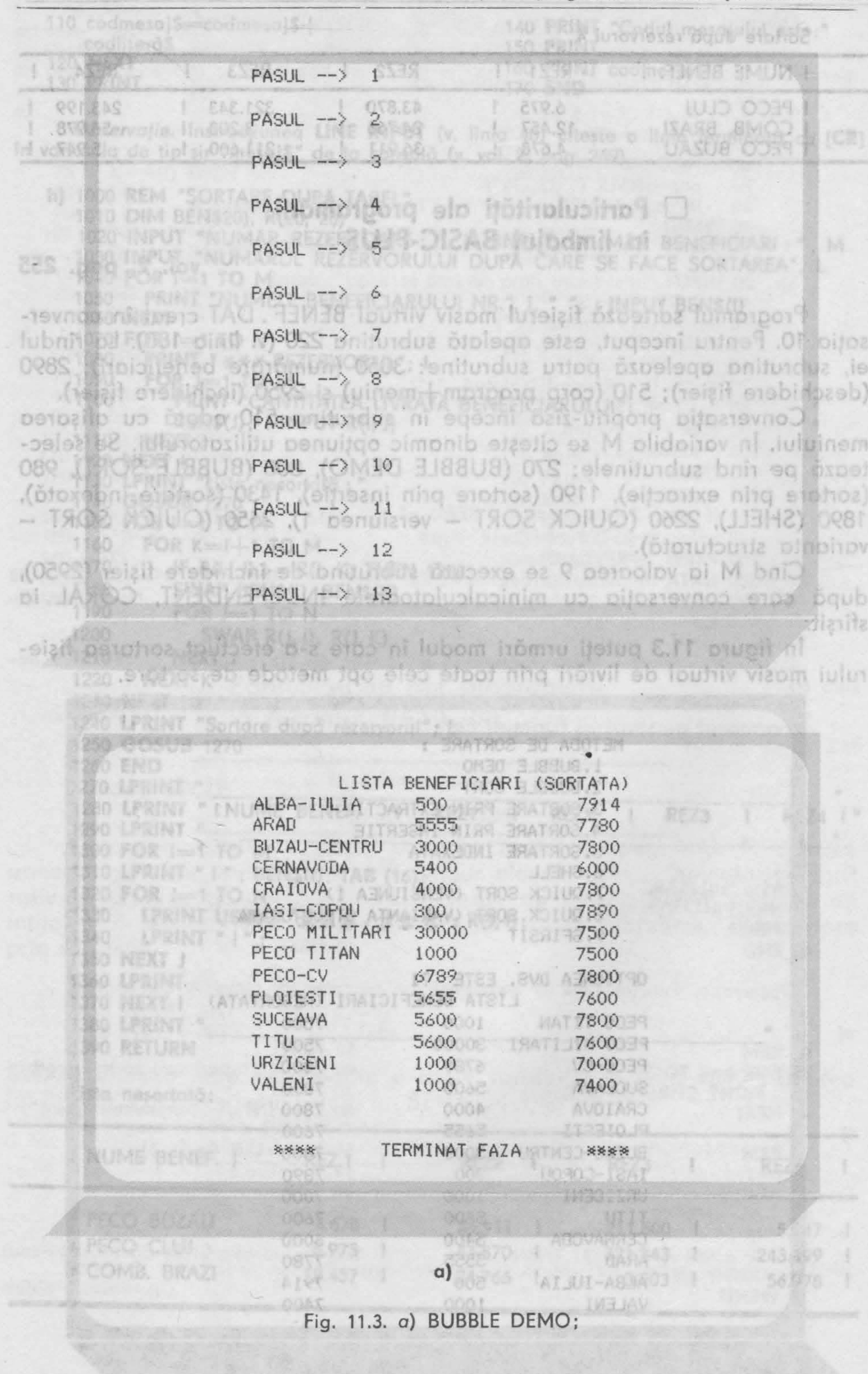
Cînd M ia valoarea 9 se execută subrutina de închidere fişier (2950), după care conversaţia cu minicalculatoarele INDEPENDENT, CORAL ia sfîrşit.

În figura 11.3 puteţi urmări modul în care s-a efectuat sortarea fişierului masiv virtual de livrări prin toate cele opt metode de sortare.



a)

Fig. 11.3. Opt metode de sortare: *a)* BUBBLE DEMO;



a)

Fig. 11.3. *a*) BUBBLE DEMO;

```
****    INCEPUT FAZA    ****

     METODA DE SORTARE :
       1.BUBBLE DEMO
       2.BUBBLE SORT
       3.SORTARE PRIN EXTRACTIE
       4.SORTARE PRIN INSERTIE
       5.SORTARE INDEXATA
       6.SHELL
       7.QUICK SORT (VERSIUNEA 1)
       8.QUICK SORT (VARIANTA STRUCTURATA)
       9.SFIRSIT

OPTIUNEA DVS. ESTE  ?3
    LISTA SORTATA :
VALENI          1000          7400
URZICENI        1000          7000
TITU            5600          7600
SUCEAVA         5600          7800
PLOIESTI        5655          7600
PECO-CV         6789          7800
PECO-TITAN      1000          7500
PECO MILITARI   30000         7500
IASI-COPOU      300           7890
CRAIOVA         4000          7800
CERNAVODA       5400          6000
BUZAU-CENTRU    3000          7800
ARAD            5555          7780
ALBA-IULIA      500           7914

    ****     TERMINAT FAZA     ****
```

b)

```
****    INCEPUT FAZA    ****

     METODA DE SORTARE :
       1.BUBBLE DEMO
       2.BUBBLE SORT
       3.SORTARE PRIN EXTRACTIE
       4.SORTARE PRIN INSERTIE
       5.SORTARE INDEXATA
       6.SHELL
       7.QUICK SORT (VERSIUNEA 1)
       8.QUICK SORT (VARIANTA STRUCTURATA)
       9.SFIRSIT

OPTIUNEA DVS. ESTE  ?2

FISIER BENEFICIARI (SORTAT)
ALBA-IULIA      500           7914
ARAD            5555          7780
BUZAU-CENTRU    3000          7800
CERNAVODA       5400          6000
CRAIOVA         4000          7800
IASI-COPOU      300           7890
PECO MILITARI   30000         7500
PECO TITAN      1000          7500
PECO-CV         6789          7800
PLOIESTI        5655          7600
SUCEAVA         5600          7800
TITU            5600          7600
URZICENI        1000          7000
VALENI          1000          7400

    ****     TERMINAT FAZA     ****
```

c)

Fig. 11.3. *b*) sortare prin extracție; *c*) BUBBLE SORT;

```
     ****   INCEPUT FAZA    ****


          METODA DE SORTARE :
            1.BUBBLE DEMO
            2.BUBBLE SORT
            3.SORTARE PRIN EXTRACTIE
            4.SORTARE PRIN INSERTIE
            5.SORTARE INDEXATA
            6.SHELL
            7.QUICK SORT (VERSIUNEA 1)
            8.QUICK SORT (VARIANTA STRUCTURATA)
            9.SFIRSIT

      OPTIUNEA DVS. ESTE  ?4
        LISTA SORTATA :
      ALBA-IULIA      500           7914
      ARAD            5555          7780
      BUZAU-CENTRU    3000          7800
      CERNAVODA       5400          6000
      CRAIOVA         4000          7800
      IASI-COPOU      300           7890
      PECO MILITARI   30000         7500
      PECO TITAN      1000          7500
      PECO-CV         6789          7800
      PLOIESTI        5655          7600
      SUCEAVA         5600          7800
      TITU            5600          7600
      URZICENI        1000          7000
      VALENI          1000          7400


        ****     TERMINAT FAZA       ****
```

d)

```
     ****   INCEPUT FAZA    ****


          METODA DE SORTARE :
            1.BUBBLE DEMO
            2.BUBBLE SORT
            3.SORTARE PRIN EXTRACTIE
            4.SORTARE PRIN INSERTIE
            5.SORTARE INDEXATA
            6.SHELL
            7.QUICK SORT (VERSIUNEA 1)
            8.QUICK SORT (VARIANTA STRUCTURATA)
            9.SFIRSIT

      OPTIUNEA DVS. ESTE  ?5
```

e)

Fig. 11.3. *d)* sortare prin inserţie; *e)* sortare indexată;

```
PENTRU CONTINUARE TASTATI (Y/N) ?Y
CIMPUL : 1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ?1
   INREGISTRARI SORTATE DUPA CIMPUL  1
ALBA-IULIA      500         7914
ARAD            5555        7780
BUZAU-CENTRU                3000        7800
CERNAVODA       5400        6000
CRAIOVA         4000        7800
IASI-COPOU      300         7890
PECO MILITARI               30000       7500
PECO TITAN      1000        7500
PECO-CV         6789        7800
PLOIESTI        5655        7600
SUCEAVA         5600        7800
TITU            5600        7600
URZICENI        1000        7000
VALENI          1000        7400
PENTRU CONTINUARE TASTATI (Y/N) ?Y
CIMPUL : 1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ?2
 INREGISTRARI SORTATE DUPA CIMPUL  2
IASI-COPOU      300         7890
ALBA-IULIA      500         7914
VALENI          1000        7400
URZICENI        1000        7000
PECO TITAN      1000        7500
BUZAU CENTRU                3000        7800
CRAIOVA         4000        7800
CERNAVODA       5400        6000
ARAD            5555        7780
TITU            5600        7600
SUCEAVA         5600        7800
PLOIESTI        5655        7600
PECO-CV         6789        7800
PECO MILITARI               30000       7500
```

```
PENTRU CONTINUARE TASTATI (Y/N) ?Y
CIMPUL : 1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ?3
INREGISTRARI SORTATE DUPA CIMPUL  3
CERNAVODA       5400        6000
URZICENI        1000        7000
VALENI          1000        7400
PECO TITAN      1000        7500
PECO MILITARI               30000       7500
TITU            5600        7600
PLOIESTI        5655        7600
ARAD            5555        7780
SUCEAVA         5600        7800
PECO-CV         6789        7800
CRAIOVA         4000        7800
BUZAU-CENTRU                3000        7800
IASI-COPOU      300         7890
ALBA-IULIA      500         7914

PENTRU CONTINUARE TASTATI (Y/N) ?N


     ****     TERMINAT FAZA      ****
```

Fig. 11.3. *e)*

```
**** INCEPUT FAZA ****

METODA DE SORTARE :
  1.BUBBLE DEMO
  2.BUBBLE SORT
  3.SORTARE PRIN EXTRACTIE
  4.SORTARE PRIN INSERTIE
  5.SORTARE INDEXATA
  6.SHELL
  7.QUICK SORT (VERSIUNEA 1)
  8.QUICK SORT (VARIANTA STRUCTURATA)
  9.SFIRSIT

OPTIUNEA DVS. ESTE ?6
   LISTA NESORTATA
VALENI   URZICENI  TITU      SUCEAVA   PLOIESTI
PECO-CV  PECO TITAN PECO MILITARI IASI-COPOU CRAIOVA
CERNAVODA BUZAU-CENTRU ARAD  ALBA-IULIA
LISTA SORTATA
ALBA-IULIA ARAD    BUZAU-CENTRU CERNAVODA CRAIOVA
IASI-COPOU PECO MILITARI PECO TITAN PECO-CV PLOIESTI
SUCEAVA  TITU     URZICENI  VALENI
 73 COMPARATII
 21 SCHIMBARI

  ****     TERMINAT FAZA     ****
```

f)

```
**** INCEPUT FAZA ****

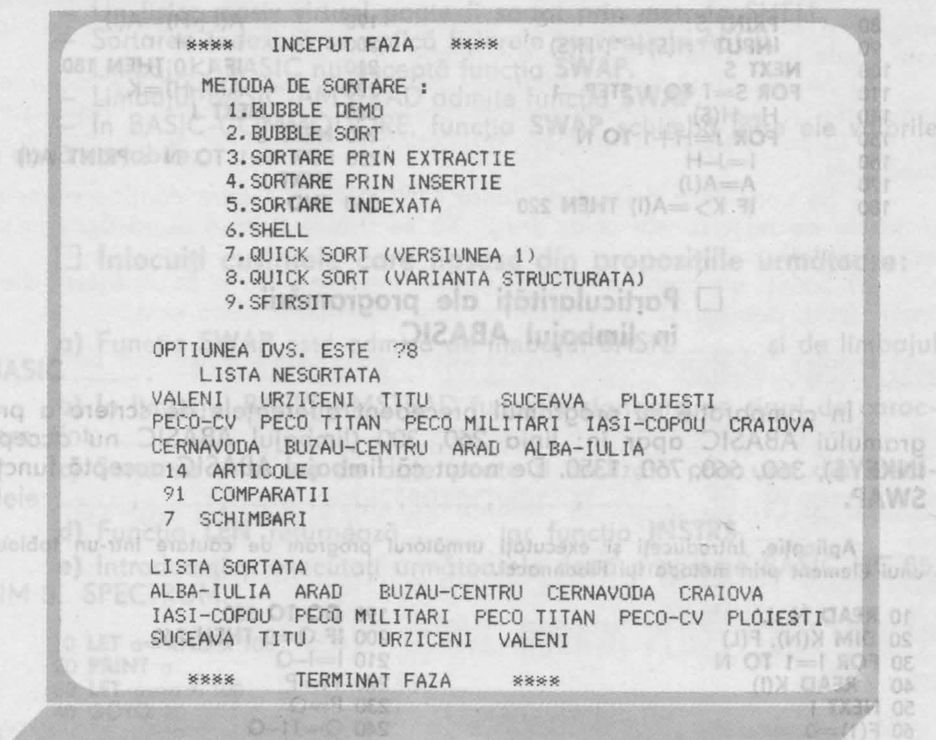
METODA DE SORTARE :
  1.BUBBLE DEMO
  2.BUBBLE SORT
  3.SORTARE PRIN EXTRACTIE
  4.SORTARE PRIN INSERTIE
  5.SORTARE INDEXATA
  6.SHELL
  7.QUICK SORT (VERSIUNEA 1)
  8.QUICK SORT (VARIANTA STRUCTURATA)
  9.SFIRSIT

OPTIUNEA DVS. ESTE ?7
   LISTA NESORTATA
VALENI   URZICENI  TITU      SUCEAVA   PLOIESTI
PECO-CV  PECO TITAN  PECO MILITARI  IASI-COPOU  CRAIOVA
CERNAVODA  BUZAU-CENTRU  ARAD  ALBA-IULIA
LISTA SORTATA
ALBA-IULIA  ARAD   BUZAU-CENTRU  CERNAVODA  CRAIOVA
IASI-COPOU  PECO MILITARI  PECO TITAN  PECO-CV  PLOIESTI
SUCEAVA  TITU      URZICENI  VALENI
 14  BENEFICIARI
 91  COMPARATII
 7  SCHIMBARI

  ****     TERMINAT FAZA ****
```

g)

Fig. 11.3. *f)* sortare SHELL; *g)* QUICK SORT (VERSIUNEA 1);



h)

Fig. 11.3. *h)* QUICK SORT (VARIANTA STRUCTURATĂ).

Aplicație. Introduceți și executați următoarele programe. Transformați-le apoi în subrutine pentru aplicațiile dvs.

a)

```
10  INPUT "N="; N
20  DIM A(N)
30  FOR I=1 TO N
40    A(I)=RND(I)
50  NEXT I
60  FOR J=2 TO N
70    I=J-1
90    A=A(J)
100   IF K>=A(I) THEN 140
110   A(I+1)=A(I)
120   I=I-1
130   IF I>0 THEN 100
140   A(I+1)=K
150 NEXT J
160 FOR I=1 TO N
170   PRINT (A(I)
180 NEXT I
```

b)

```
10 INPUT "N="; N
20 DIM A(N)
30 FOR I=1 TO N
40   A(I)=RND(I)
50 NEXT I
60 FOR J=N TO 2 STEP -1
70   B=A(J)
80   L=J
90   FOR R=J TO 1 STEP -1
100    IF A(R)<=B THEN 130
110    L=R
120    B=A(R)
130  NEXT R
140  D=A(J)
150  A(J)=A(L)
160  A(L)=D
170 NEXT J
180 FOR I=1 TO N
190   PRINT A(I)
200 NEXT I
210 END
```

c)

```
10 INPUT "N="; N
20 DIM A(N)
30 FOR I=1 TO N
40   A(I)=RND(1)
50 NEXT I
60   INPUT "T="; T
70   FOR S=T TO 1 STEP -1
```

```
80      PRINT S;
90      INPUT "H(S)="; H(S)
100   NEXT S
110   FOR S=T TO 1 STEP -1
140      H=H(S)
150      FOR J=H+1 TO N
160         I=J-H
170         A=A(J)
180         IF K>=A(I) THEN 220
190            A(I+H)=A(I)
200            I=I-H
210            IF I>0 THEN 180
220            A(I+H)=K
230      NEXT J
240 NEXT S
250 FOR I=1 TO N : PRINT A(I) :
    NEXT I
260 END
```

□ Particularități ale programării în limbajul ABASIC

În comparație cu programul precedent diferențele de scriere a programului ABASIC apar in: linia 260, 300 (limbajul ABASIC nu acceptă **INKEY$**), 360, 660, 760, 1350. De notat că limbajul ABASIC acceptă funcția **SWAP.**

Aplicație. Introduceți și executați următorul program de căutare într-un tablou a unui element prin metoda lui Fibonnacci.

```
10 READ N, L
20 DIM K(N), F(L)
30 FOR I=1 TO N
40   READ K(I)
50 NEXT I
60 F(1)=0
70 F(2)=0
80 FOR I=3 TO L
90 F(I)=F(I-1)+F(I-2)
100 NEXT I
110 INPUT "NR. CĂUTAT"; K
120 I=F(L-1)
130 P=F(L-2)
140 Q=F(L-3)
150 IF K<K(I) THEN 200
160 IF>K(I) THEN 280
170 PRINT I, K(I)
180 PRINT
190 GO TO 330
200 IF Q=0 THEN 260
210 I=I-Q
220 T1=P
230 P=Q
240 Q=T1-Q
250 GOTO 150
260 PRINT
270 GOTO 330
280 IF P=1 THEN 260
290 I=I+Q
300 P=P-Q
310 Q=Q-P
320 GOTO 150
330 END
340 DATA 12, 8
350 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
    11, 12
```

TEMA 11

□ Răspundeți prin DA sau NU la următoarele întrebări:

– Funcția **LEN(Y$)** returnează lungimea șirului de caractere Y$.
– În instrucțiunea

```
10 IF INSTR(Y$, X$)=0 THEN 200
```

funcția **INSTR** caută prima apariție a șirului X$ în șirul Y$ și returnează poziția de la care începe corespondența.

– Un fişier masiv virtual poate fi sortat prin metoda SHELL.
– Sortarea indexată se aplică fişierele secvenţiale ASCII.
– Limbajul ABASIC nu acceptă funcţia **SWAP**.
– Limbajul BASIC-AMSTRAD admite funcţia **SWAP**.
– În BASIC-COMMODORE, funcţia **SWAP** schimbă între ele valorile a două variabile.

☐ Înlocuiţi cuvintele care lipsesc din propoziţiile următoare:

a) Funcţia **SWAP** este admisă de limbajul BASIC .... şi de limbajul BASIC ....

b) În limbajul BASIC-AMSTRAD funcţiile standard pe şiruri de caractere sînt ....

c) Sortarea unui set de date poate fi realizată prin una din metodele: ...., ..., ..., ..., ..., ...

d) Funcţia **LEN** returnează .... iar funcţia **INSTR$** ....

e) Introduceţi şi executaţi următoarele două programe BASIC HC-85, TIM S, SPECTRUM.

```
10 LET a=RND * 100
20 PRINT a
30 LET a=a * 100
40 GOTO 20
```

şi

```
10 LET a=RND * 100
20 PRINT TAB 15 - LEN STR$ INT a; a
30 LET a=a * 10
40 GO TO 20
```

În timp ce primul program afişează ...., cel de-al doilea program afişează ....

f) Care va fi rezultatul execuţiei următorului program BASIC–SPECTRUM?

```
10 PRINT 2
20 PRINT STR$ 2
30 PRINT 1/3
40 PRINT STR$ (1/3)
50 PRINT 9E15
60 PRINT STR$ 9E15
```

☐ Scrieţi cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Se consideră fişierul secvenţial UTILAJ care conţine numele şi notele de la examenul de Programare ale studenţilor din grupa 1403. Să se listeze fişierul în ordine alfabetică şi după notele obţinute. Se va obţine, de asemenea, o listă care să aibă structura: număr curent, nume student, notă,

rang. Prin rang se înțelege locul studentului în grupă, după nota sa; la note egale se păstrează ordinea din datele inițiale.

b) Se consideră fișierul dicționar ARMĂTURI – care conține codurile, denumirile și prețul produselor (armăturilor) fabricate de I.U.P. Tîrgoviște. Să se listeze fișierul în ordinea alfabetică a produselor și descrescător după preț.

c) Se consideră fișierul dicționar ONOMASTICI care conține numele și datele de naștere ale celor dragi. Să se listeze fișierul după data nașterii prietenilor.

d) Scrieți un program BASIC pentru interclasarea a două fișiere dicționar (cod, denumire) secvențiale sortate ascendent după cod.

Indicație. Se va apela subrutina de interclasare (prezentată mai jos) pentru doi vectori X și Y ce conțin m, respectiv n elemente.

```
50  DIM X(M), Y(N), Z(M+N)
100 I=1
110 J=1
120 K=1
130 IF X(I)>Y(J) THEN 220
140 Z(K)=X(I)
150 K=K+1
160 I=I+1
170 IF I<=M THEN 130
180 FOR K=J TO N
190   Z(K+M)=Y(K)
200 NEXT K
210 GOTO 290
220 Z(K)=Y(J)
230 K=K+1
240 J=J+1
250 IF J<=N THEN GOTO 130
260 FOR K=1 TO M
270   Z(K+N)=X(K)
280 NEXT K
290 FOR I=1 TO M+N
300   PRINT Z(I)
310 NEXT I
320 RETURN
```

☐ **Să se proiecteze și să se realizeze un program BASIC** ghidat de un meniu pentru sortarea fișierului secvențial JUCĂRII.DAT (v. Conversația 7) după cod, utilizînd subrutinele BUBBLE SORT și SHELL (v. Conversația 11).

☐ **Să se proiecteze și să se realizeze un program BASIC-PLUS** ghidat de un meniu pentru sortarea fișierului masiv virtual BENEF.DAT (v. Conversația 10) după cod proveniență și preț de achiziție, utilizînd subrutinele pentru sortarea indexată, extracție și inserție.

CONVERSAȚIA 12

Reprezentarea livrărilor de lichide din trei rezervoare cu ajutorul histogramelor. Tehnici de trasare a histogramelor în 2D și 3D. Programarea culorilor. Instrucțiunile grafice WINDOW, VIEWPORT, MOVE, DRAW, RMOVE, RDRAW, PLOT, POINT, CIRCLE. APLICAȚII și TESTE pentru cititor



□ Elemente de grafică interactivă

Nu știu alții cum sînt*, dar eu cînd mă gîndesc la orele de desen și la planșele (la scară și trase-n tuș) pe care trebuia să le arăt a doua zi profesorului mi se umple și-acum sufletul de neliniște. Neîndemînarea mea, lipsa de talent și sigur lipsa de efort susținut m-au ținut întotdeauna departe de blocul de desen, și mai tîrziu de planșeta de la întreprindere. Din momentul cînd am descoperit că prietenul meu calculatorul știe să și deseneze vă mărturisesc că neliniștea a fost și mai mare dar de această dată bucuriile n-au întîrziat să apară.

Din dorința de a vă implica și pe dumneavoastră, stimați cititori în conversațiile de grafică cu calculatorul (aMIC, PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE, M118, TPD, JUNIOR) vă propunem să faceți o încercare desenînd pentru început figuri geometrice simple, după care veți vedea și singuri ce va urma.

Nu vă alarmați! Și noi am fost în situația dvs. nu cu mult timp în urmă. Vă vom explica tot ceea ce doream și noi să știm atunci cînd am început să desenăm cu calculatorul. Nu putem promite că veți cuceri universul după o jumătate de oră de conversație dar putem garanta că atunci cînd prietenii dvs. vă vor întreba cum le-ați luat-o înainte cu calculatorul personal îi veți uimi cu desenele, graficele ori animațiile realizate cu propriul efort.

Rezoluție, pixel, memorie ecran

În mod grafic, ecranul de vizualizare al calculatorului dvs. este „decupat" în puncte pe care le puteți aprinde sau stinge cu instrucțiuni grafice consacrate. Totalitatea punctelor distincte ce pot fi reprezentate pe un ecran (dispozitiv grafic) reprezintă **spațiul grafic.** Calitatea imaginilor obținute (pe ecran) depinde de ceea ce este cunoscut sub numele de **rezoluție** care este un atribut al spațiului grafic.

Rezoluția se măsoară prin numărul de puncte orizontale și verticale ce pot fi individualizate. Astfel, o rezoluție de 256×176 arată că există 256 de puncte pe axa orizontală a ecranului și 176 de puncte pe axa verticală.

În egală măsură este utilizat și termenul de **pixel** (prescurtare de la "picture elements"). De exemplu, ecranul TV, din punct de vedere al calculatorului PRAE este împărțit în 65 356 pixeli (256×256 puncte).

Majoritatea ecranelor pentru calculatoarele personale sînt dotate cu circuite de comandă compuse din **memorii ecran** sau **memorii video** (separate de memoria de lucru) de regulă, de tip RAM (Random Acces Memory).

* Ne amintim cu emoțiile copilăriei de marele nostru povestitor, Ion Creangă.

Utilizarea ferestrei şi a vizorului

Instrucţiunile grafice . . . consacrate referă puncte ale spaţiului grafic prin coordonatele lor (X, Y), exprimate în unităţi de măsură naturale, corespunzătoare mărimilor reprezentate grafic (newtoni, amperi, secunde etc.). Totalitatea valorilor de mai sus reprezintă **"spaţiul utilizator"** sau virtual.

Observaţie. Spaţiul virtual este limitat numai de precizia aritmeticii maşinii (1E–63, 8E+68) urmînd a fi reprezentat la o anumită scară, în cadrul spaţiului grafic (pe suprafaţa display-ului).

Datorită faptului că spaţiul ecranului este mult mai mic decît spaţiul pe care îl are la dispoziţie utilizatorul (desenatorul), se afişează pe ecran numai o parte din desen, la o scară care să permită o precizie acceptabilă. Această "fereastră" (WINDOW) permite analiza în detaliu a porţiunii alese din desen. În acest caz se au în vedere două probleme distincte: a) găsirea transformării care trebuie aplicate desenului iniţial pentru a fi vizualizată pe ecran porţiunea dorită (WINDOWING); b) identificarea elementelor din desenul iniţial care sînt vizibile în fereastră (CLIPPING).

Observaţie. Fereastra (porţiunea din desen care trebuie afişată) şi vizorul (zona de pe ecran pe care se face afişarea) sînt două dreptunghiuri asemenea.

De notat că elementele care definesc fereastra (patru argumente) sînt date în unităţi de utilizator. În funcţie de conţinutul desenului iniţial, ele pot fi: ani, km, lei etc. De multe ori, afişarea datelor cuprinse în fereastră nu ocupă întregul ecran, ci numai o parte a acestuia, în vederea plasării unor titluri, texte, indicaţii sau grafice suplimentare. Zona din ecran pe care va fi proiectat conţinutul ferestrei o vom numi vizor. Elementele care definesc vizorul (patru argumente) sînt date în unităţile sistemului grafic utilizat.

Observaţie. Unitatea în care se exprimă limitele elementelor ce definesc vizorul s-a ales, ca fiind egală cu 1% din latura pătratului cel mai mare, ce poate fi înscris în suprafaţa ecranului (unitate grafică sau prescurtat U.G.).

Înalta şi joasa rezoluţie

Datorită diferenţei de rezoluţie de la calculator la calculator şi datorită faptului că fiecare sistem de calcul dispune de anumite facilităţi privind reprezentarea grafică se foloseşte şi noţiunea de înaltă şi joasă rezoluţie. De exemplu, graficele alcătuite din segmente de dreaptă au o înaltă rezoluţie pe cînd cele alcătuite din caractere semigrafice, de utilizator au o joasă rezoluţie.

Sisteme de coordonate

Sistemul de coordonate carteziene. Cum o foaie de hîrtie nu are decît două dimensiuni (lungime şi lăţime) sînt suficiente două numere pentru a identifica toate punctele de pe suprafaţa acestei foi. Într-un sistem de coordonate carteziene xoy un punct M (v. figura 12.1) este reperat prin coordonatele X=4 şi Y=3. Fiecare din axele OX şi OY definesc atît direcţia cît şi

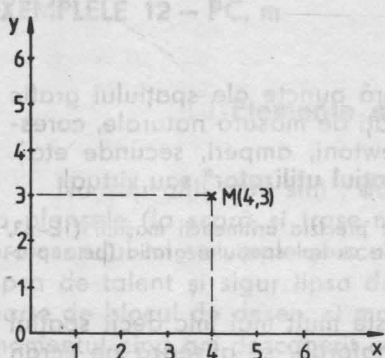


Fig. 12.1.

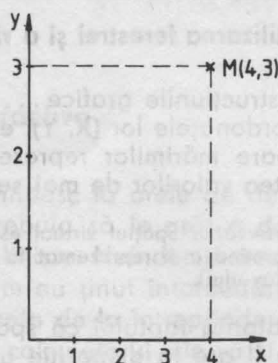


Fig. 12.2.

o unitate de măsură (v. figura 12.1 în care axele sînt perpendiculare şi au aceeaşi unitate de măsură). Nu întotdeauna lucrurile stau la fel (v. fig. 12.2 şi 12.3).

Sistemele frecvent utilizate (cu coordonate carteziene) sînt cele cu axele perpendiculare fără a avea însă mereu aceeaşi unitate de măsură pentru cele două axe (v. fig. 12.4 şi 12.5).

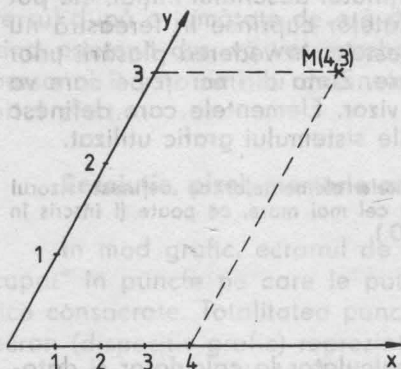


Fig. 12.3.

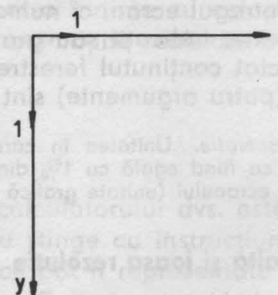


Fig. 12.4.

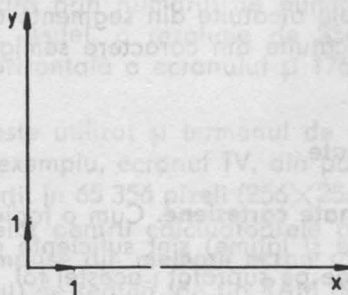


Fig. 12.5.

Sisteme de coordonate polare. Un punct M (x, y) poate fi identificat (într-un sistem de coordonate polare) printr-un unghi (θ) şi o lungime (ρ). În figura 12.6 este reprezentat un astfel de sistem.

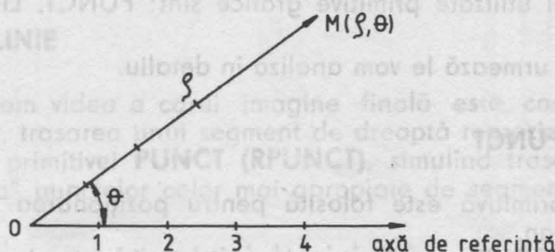


Fig. 12.6.

Trecerea de la un sistem de coordonate la altul. Să presupunem că: a) axa x, în coordonate carteziene se confundă cu axa de referinţă pentru coordonate polare; b) axele ox şi oy sînt perpendiculare şi au o unitate de măsură comună; c) originea axelor este comună.

În acest caz, punctul M (v. fig. 12.7) are coordonatele:

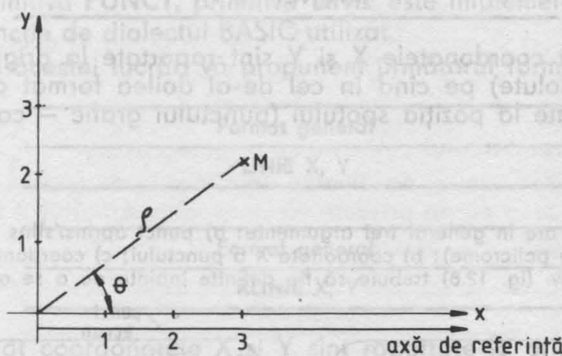


Fig. 12.7.

X=3; Y=3 (coordonate carteziene)

ρ=3,5; θ=60° (coordonate polare).

Trecerea de la un sistem de referinţă la altul se obţine prin relaţiile:

$X=\rho \cos \theta$ (1) sau

$Y=\rho \sin \theta$ (2) $\theta=\text{arcs sin}\left(\frac{Y}{\sqrt{x^2+y^2}}\right)$ (5)

$\rho=\sqrt{x^2+y^2}$ (3) sau

$\theta=\text{arc cos}\left(\frac{x}{\sqrt{x^2+y^2}}\right)$ (4) $\theta=\text{arc tg}\left(\frac{Y}{X}\right)$ (6)

☐ Primitive grafice

Pentru a desena cu ... calculatorul, apare ca necesară cunoaşterea şi înţelegerea unor primitive (funcţii) mult apropiate de limbajul matematic clasic. Cele mai utilizate primitive grafice sînt: PUNCT, LINIE, CERC, PĂTRAT etc.

În cele ce urmează le vom analiza în detaliu.

Primitiva PUNCT

Această primitivă este folosită pentru poziţionarea spotului într-un punct de pe ecran.

Primitiva **PUNCT** are diverse denumiri în funcţie de dialectul BASIC pe care-l utilizaţi.

În ceea ce ne priveşte vă propunem următorul format general:

Format general
PUNCT (X, Y)

Format general
RPUNCT (X, Y)

În primul format coordonatele X şi Y sînt raportate la originea sistemului (coordonate absolute) pe cînd în cel de-al doilea format coordonatele X şi Y sînt raportate la poziţia spotului (punctului grafic – coordonate relative).

Remarci

- Primitiva **PUNCT** are în general trei argumente: a) punct aprins/stins (sau culoare pentru sisteme policrome); b) coordonata X a punctului; c) coordonata Y a punctului.
- Cele trei valori (v. fig. 12.8) trebuie să fie definite înainte de a se apela primitiva.

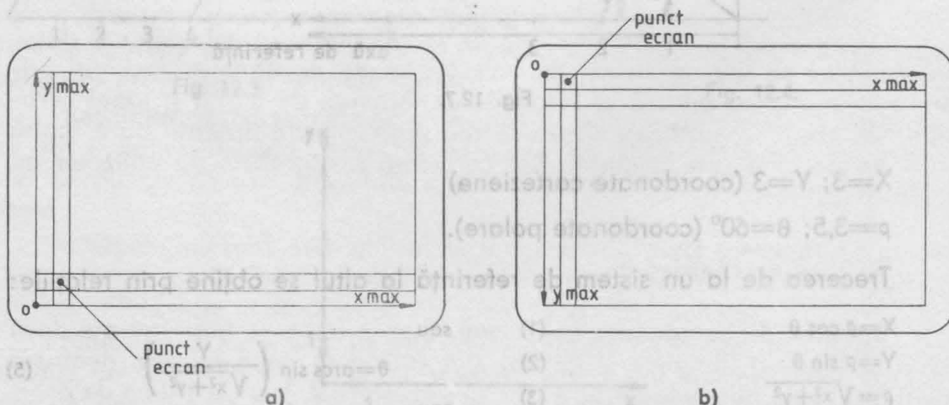


Fig. 12.8. Primitiva **PUNCT**: *a)* originea axelor în colţul din stînga jos; *b)* originea axelor în colţul din stînga sus.

- Valorile lui X şi Y trebuie să satisfacă inegalităţile:

  0⩽X⩽Xmax sau numărul maxim de puncte adresabile în direcţia OX;

  şi

  0⩽Y⩽Ymax sau numărul maxim de puncte adresabile în direcţia OY.

Primitiva LINIE

Pe un sistem video a cărui imagine finală este constituită dintr-o reţea de puncte, trasarea unui segment de dreaptă reprezintă o succesiune de apeluri ale primitivei **PUNCT (RPUNCT)**, simulînd traseul segmentului prin "aprinderea" punctelor celor mai apropiate de segmentul adevărat (v. figura 12.9).

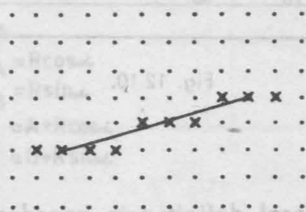


Fig. 12.9.

Ca şi primitiva **PUNCT**, primitiva **LINIE** este implementată sub diverse denumiri în funcţie de dialectul BASIC utilizat.

În cadrul acestei lucrări vă propunem următorul format general:

Format general
LINIE X, Y

Format general
RLINIE X, Y

În primul format coordonatele X şi Y sînt raportate la originea sistemului. În cel de-al doilea format coordonatele X şi Y sînt relative (fiind raportate la poziţia spotului la momentul execuţiei primitivei – instrucţiunii).

Aplicaţie. Scrieţi o secvenţă algoritmică (în pseudocod) pentru trasarea liniei din fig. 12.10.

```
AB   SEQ
     PUNCT (20, 20)
     LINIE 40, 40
AB   END
```

sau

```
AB   SEQ
     RPUNCT (20, 20)
     RLINIE 20, 20
AB   END
```

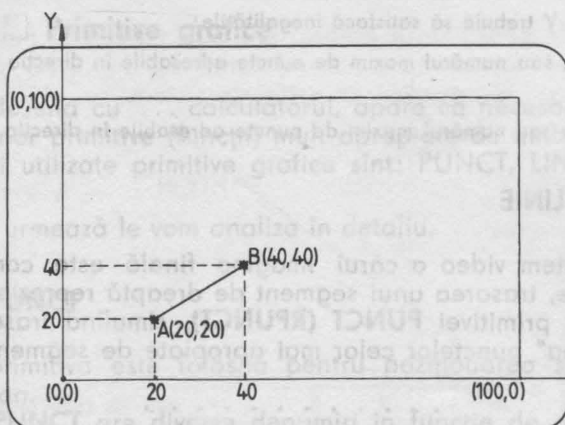


Fig. 12.10.

Primitiva CERC

Un cerc este în general definit prin coordonatele centrului şi lungimea razei. De nenumărate ori ecuaţia sa

$$x^2+y^2-2ax-2by+c=0 \quad (7)$$

$$(\text{cu } R^2=a^2+b^2-c^2)$$

se restrînge în:

$$x^2+y^2-R^2=0 \quad (8)$$

atunci cînd centrul cercului coincide cu originea axelor.

Primitiva **CERC** are şi ea diverse denumiri în dialectele BASIC cu facilităţi grafice. În ceea ce ne priveşte vă propunem următorul format general:

Format general
CERC X, Y, R

unde X, Y reprezintă coordonatele centrului iar R este raza cercului.

Aplicaţie. Să se deseneze un cerc utilizind primitivele grafice **PUNCT** şi **LINIE** (v. figura 12.11).

Procedura este următoarea. Vom aproxima arcul cu coarda generind arce mici (de exemplu $\frac{PI}{10}$ radiani) după care le vom uni.

```
CERC   SEQ
           INPUT A, B, R
           PUNCT (A+R, B)
ARC    PENTRU I DE LA 0 LA 2 * PI PAS π/10
           LINIE (A+R * COS(I), B+R * SIN (I))
ARC    SFÎRŞIT
CERC   END
```

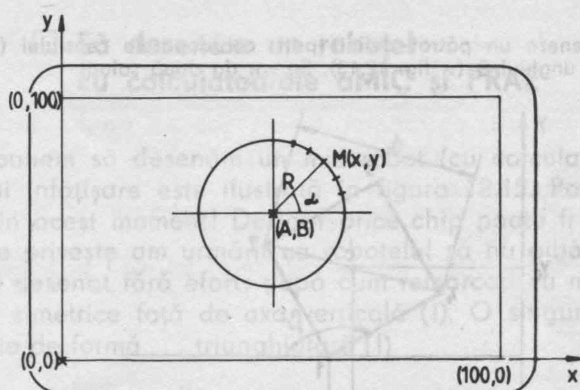


Fig. 12.11.

Primitiva PĂTRAT

Un pătrat este în general definit prin coordonatele (x, y) centrului, lungimea laturii (A) şi unghiul α (ALFA) format de o latură cu direcţia axei OX (v. fig. 12.12).

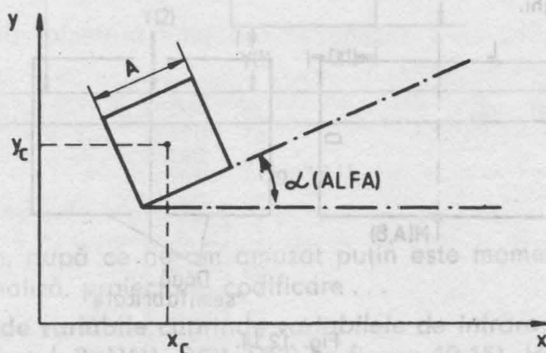


Fig. 12.12.

În cadrul acestei lucrări vom folosi următorul format general

Format general
PĂTRAT X, Y, A, ALFA

în care X, Y, A şi ALFA au semnificaţia din figura 12.12.

TEST

Să se deseneze un pătrat definit prin: coordonatele centrului (Xc, Yc), lungimea diagonalei (d) și unghiul β (v. fig. 12.13). Se vor da două soluții.

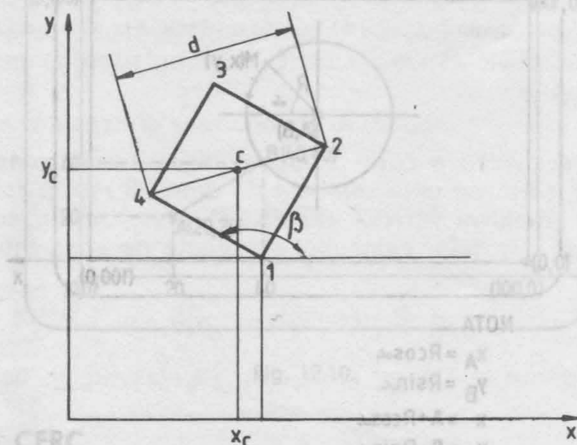


Fig. 12.13.

Aplicație. Următorul subalgoritm

```
DR      SEQ
              PUNCT (A, B)
              LINIE (A+L/2, B)
              LINIE (A+L/2, B+D)
              LINIE (A, B+D)
DR      END
```

desenează o jumătate de dreptunghi (v. figura 12.14). Pornind de la acest subalgoritm desenați un dreptunghi.

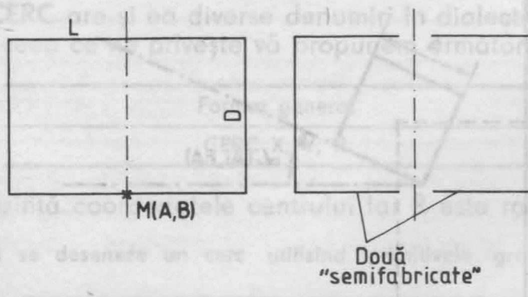


Fig. 12.14.

```
DREPT   SEQ
        INPUT A, B, D
D       PENTRU I DE LA 1 LA 3 PAS 2
          PUNCT (A, B)
          LINIE (A+J-2) * L/2, B)
          LINIE (A+J-2) * L/2, B+D)
          LINIE (A, B+D)
D       SFÎRȘIT
DREPT   END
```

□ Să desenăm un roboţel cu calculatoarele aMIC şi PRAE

Ne propunem să desenăm un mic robot (cu calculatoarele aMIC şi PRAE) a cărui înfăţişare este ilustrată în figura 12.15. Poate mulţi dintre dvs. zîmbesc în acest moment! Desigur orice chip poate fi şi înfrumuseţat! În ceea ce ne priveşte am urmărit ca roboţelul să nu aibă capul "pătrat", şi să poată fi desenat fără efort; după cum remarcaţi cu multă "risipă" de dreptunghiuri simetrice faţă de axa verticală (!). O singură excepţie face nasul care este de formă . . . triunghiulară (!)

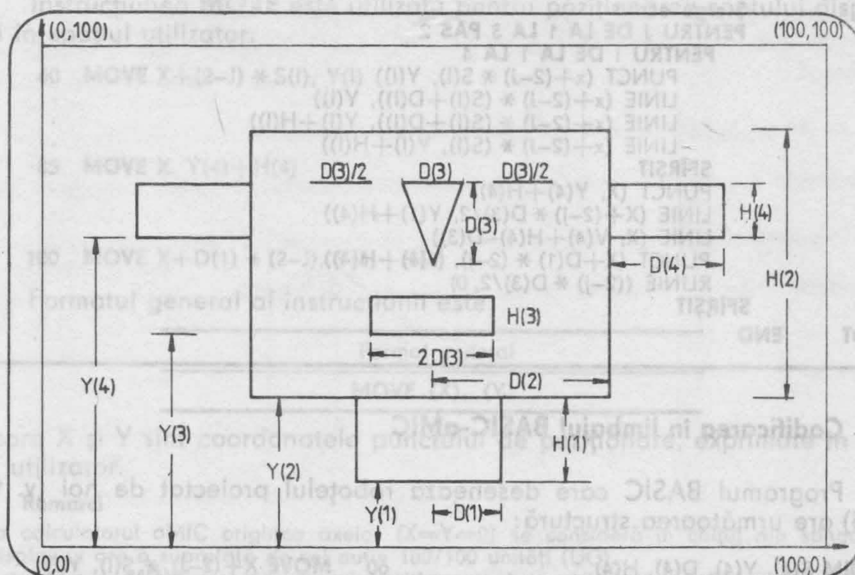


Fig. 12.15.

Şi-acum, după ce ne-am amuzat puţin este momentul să ne apucăm de treabă: analiză, proiectare, codificare . . .

Tabela de variabile cuprinde *variabilele de intrare:* X (abscisa axei de simetrie a robotului), D(1), D(2), D(3) (v. figura 12.15), H(1), H(2), H(3), H(4) (v. figura 12.15), Y(1), Y(3), Y(4) (v. figura 12.15), *variabilele de stare:* S (vector de lucru), I, J (variabile de control) şi *variabilele de ieşire* (Y şi H).

Variabilele de intrare primesc în timpul execuţiei programului următoarele valori: X=50; D(1)=15; H(1)=20; Y(1)=20; D(2)=25; H(2)=40; D(3)=10; H(3)=5; Y(3)=50; H(4)=10; Y(4)=60.

Pseudocodul pentru desenarea roboţelului este prezentat în tabelul 12.1.

Tabelul 12.1

PSEUDOCODUL

```
ROBOT    SEQ
           INPUT X
A          PENTRU I DE LA 1 LA 4
             INPUT D(I)
             INPUT H(I)
A1           DACĂ I≠2
               INPUT Y(I)
A1           SFÎRŞIT
A          SFÎRŞIT
           Y(2)=Y(1)+H(1)
           S(4)=D(2)
           Şterge ecran
B          PENTRU J DE LA 1 LA 3 PAS 2
B1           PENTRU I DE LA 1 LA 4
               PUNCT (x+(2-J) * S(I), Y(I))
               LINIE (x+(2-J) * (S(I)+D(I)), Y(I))
               LINIE (x+(2-J) * (S(I)+D(I)), Y(I)+H(I))
               LINIE (x+(2-J) * (S(I), Y(I)+H(I))
B1           SFÎRŞIT
             PUNCT (X, Y(4)+H(4))
             LINIE (X+(2-j) * D(3)/2, Y(1)+H(4))
             LINIE (X, V(4)+H(4)-D(3))
             PUNCT (X+D(1) * (2-j), Y(4)+H(4))
             RLINIE ((2-j) * D(3)/2, 0)
B          SFÎRŞIT
ROBOT    END
```

Codificarea în limbajul BASIC-aMIC

Programul BASIC care desenează roboţelul proiectat de noi (v. fig. 12.15) are următoarea structură:

```
5 DIM S(4), Y(4), D(4), H(4)
7 PRINT "X=";
8 INPUT X
15 FOR I=1 TO 4
22   PRINT "D(" ; I ; ")=" ;
23   INPUT D(I)
24   PRINT "H(" ; I ; ")=";
25   INPUT H(I)
26   IF I=2 THEN 30
27   PRINT "Y (" ; I ; ")=" ;
30 NEXT I
35 Y(2)=Y(2)+H(1)
40 S(4)=D(2)
45 INIT
50 FOR J=1 TO 3 STEP 2
55   FOR I=1 TO 4
60     MOVE X+(2-J) * S(I), Y(I)
65     DRAW X+(2-J) * (S(I)+D(I)), Y(I)
70     DRAW X+(2-J) * (S(I)+D(I)),
       Y(I)+H(I)
75     DRAW X+(2-J) * S(I), Y(I)+H(I)
80   NEXT I
85   MOVE X, Y(4)+H(4)
90   DRAW X+(2-J) * D(3)/2,
     Y(4)+H(4)
95   DRAW X, Y(4)+H(4)-D(3)
100  MOVE X+D(1) * (2-J), Y(4)+H(4)
105  RDRAW (2-J) * D(3)/2, 0
110 NEXT j
115 STOP
120 END
```

După cum aţi putut constata el conţine trei elemente de noutate: instrucţiunile **MOVE** (v. liniile 60, 85, 100), **DRAW** (v. liniile 65, 70, 75, 90, 95) şi **RDRAW** (v. linia 105). Cele trei instrucţiuni la care se vor mai adăuga şi altele **(WINDOW, VIEWPORT)** formează setul minim de instrucţiuni gra-

fice ale limbajului BASIC-aMIC. Ele codifică primitivele **PUNCT** respectiv **LINIE** şi **RLINIE** cu care am făcut cunoştinţă nu cu mult timp în urmă. Revenind la program remarcaţi cum pentru J=1 se desenează partea stîngă a roboţelului, după cum urmează (v. liniile 55–80): jumătatea stîngă de gît (I=1), jumătatea stîngă de cap (I=2), jumătatea dreaptă de gură (I=3), urechea stîngă (I=4), jumătatea stîngă de nas (v. liniile 85–95), ochiul stîng (v. liniile 100–105).

Pentru j=2, se desenează cu aceleaşi instrucţiuni partea dreaptă a roboţelului.

Vom analiza în cele ce urmează fiecare instrucţiune grafică în parte.

MOVE

Instrucţiunea **MOVE** este utilizată pentru poziţionarea spotului display-ului în spaţiul utilizator.

```
60  MOVE X+(2–J) * S(I), Y(I)
.
.
85  MOVE X, Y(4)+H(4)
.
.
100 MOVE X+D(1) * (2–J), Y(4)+H(4)
```

Formatul general al instrucţiunii este:

Format general
MOVE ⟨X⟩, ⟨Y⟩

în care X şi Y sînt coordonatele punctului de poziţionare, exprimate în unităţi utilizator.

Remarci

- La calculatorul aMIC originea axelor (X=Y=0) se consideră în colţul din stînga jos.
- Display-ul are o suprafaţă de cel puţin 100/100 unităţi (UG).
- Instrucţiunea **MOVE** execută numai poziţionarea în punctul de coordonate (X, Y) fără a marca punctul respectiv.
- Coordonatele (X, Y) pot fi constante, variabile sau expresii. Ele sînt raportate la originea sistemului.
- Instrucţiunea **RMOVE (MOVE** Relativ) al cărei format general este acelaşi consideră coordonatele (X, Y) raportate la poziţia spotului (coordonate relative).

DRAW

Instrucţiunea **DRAW** este folosită pentru trasarea unei drepte între punctul în care se află spotul în momentul execuţiei instrucţiunii şi punctul de coordonate (X, Y) specificat.

```
65 DRAW X+(2–J) * (S(I)+D(I)), Y(I)
70 DRAW X+(2–J) * (S(I)+D(I)), Y(I), Y(I)+H(I)
.
.
90 DRAW X+(2–J) * D(3)/2, Y(4)+H(4)
95 DRAW X, Y(4)+H(4)–D(3)
```

Formatul general al instrucţiunii este:

Format general
DRAW ⟨X⟩, ⟨Y⟩

în care ⟨X⟩ şi ⟨Y⟩ precizează capătul final al liniei. Ele pot fi constante, variabile sau expresii.

Observaţie. Instrucţiunea **DRAW** codifică în limbajul BASIC–aMIC primitiva **LINIE.**

Aplicaţie. Să se traseze graficul de producţie al întreprinderii REMORCA pentru trimestrul II al anului în curs (v. figura 12.16).

Indicaţie. Se citesc în vectorul P valorile producţiei pe lunile martie, aprilie, mai, iunie: P(1), P(2), P(3), P(4). Se poziţionează spotul în punctul de coordonate X=3, Y=P(1). În cadrul unei bucle **FOR–NEXT** se unesc printr-o dreaptă cele patru puncte ale graficului.

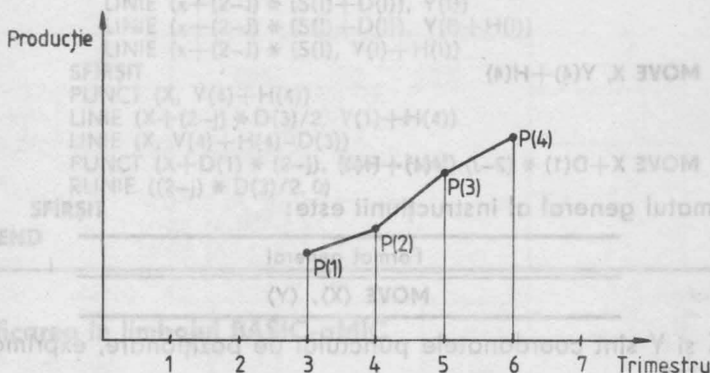


Fig. 12.16.

RDRAW

Instrucţiunea **RDRAW (DRAW** relativ) se deosebeşte de instrucţiunea **DRAW** prin faptul că coordonatele X, Y nu sînt raportate la originea sistemului, ci la poziţia spotului la momentul execuţiei instrucţiunii.

```
105 RDRAW (2–J) * D(3)/2, 0
```

Formatul general al instrucţiunii este:

Format general
RDRAW ⟨X⟩, ⟨Y⟩

Remarci

- Instrucţiunea **RDRAW** codifică în limbajul BASIC-aMIC primitiva **RLINIE.**
- Coordonatele (relative) X, Y pot fi constante, variabile sau expresii.
- Instrucţiunile **RMOVE** şi **RDRAW** se folosesc în special pentru translaţia figurilor.

Aplicație. Să se traseze un pătrat cu latura de 10 UU avînd colțul din stînga jos în origine. Scrieți un program BASIC-aMIC care să translateze pătratul în diferite puncte din spațiul utilizatorului.

WINDOW și VIEWPORT

În limbajul BASIC–aMIC "spațiul utilizator" se declară cu instrucțiunea **WINDOW** al cărei format general este:

Format general
WINDOW ⟨e1⟩, ⟨e2⟩, ⟨e3⟩, ⟨e4⟩

Remarci

- ⟨e1⟩, ⟨e2⟩, ⟨e3⟩, ⟨e4⟩ definesc un spațiu dreptunghiular (fereastră) care reprezintă limitele "spațiului utilizator", respectiv limita stîngă (e1), limita dreaptă (e2), limita inferioară (e3), limita superioară (e4). Ele pot fi constante, variabile, sau expresii.
- Instrucțiunea realizează automat trecerea de la puncte de coordonate utilizator la puncte ecran.
- Se recomandă ca laturile ferestrei declarate în instrucțiunea **WINDOW** să fie cît mai apropiate de domeniul de valori de reprezentat.
- Spațiul utilizator definit de **WINDOW** se plasează de obicei pe întreaga suprafață a ecranului.
- Instrucțiunea **WINDOW** se plasează la începutul programului (subprogramului) de reprezentare grafică.
- În absența instrucțiunii **WINDOW,** calculatorul aMIC stabilește fereastra implicit la valorile 0, 100, 0, 100.

TEST

Precizați rolul următoarelor instrucțiuni:

```
10 WINDOW - 30, 130, -30, 130
20 WINDOW - 6000, 6000, -1000, 1000
```

Nu puține sînt cazurile cînd dorim ca fereastra să fie proiectată numai pe o parte a suprafeței ecranului, cealaltă parte fiind necesară altor scopuri. În acest caz vom utiliza instrucțiunea **WIEWPORT,** al cărei format general este:

Format general
VIEWPORT ⟨e1p⟩, ⟨e2p⟩, ⟨e3p⟩, ⟨e4p⟩

unde, ⟨e1p⟩ – ⟨e4p⟩ reprezintă limitele fizice ale zonei din suprafața display-ului în aceeași ordine ca și la instrucțiunea **WINDOW.**

Remarci

- Unitatea de măsură în care se exprimă ⟨e1p⟩ – ⟨e4p⟩ s-a ales ca fiind un procent din latura pătratului cel mai mare, care poate fi înscris în suprafața display-ului. Este numită unitate grafică (UG).
- În absența instrucțiunii **VIEWPORT** calculatorul va executa implicit

```
10 VIEWPORT 0, 100, 0, 100
```

- Cu iniţializările implicite

```
10 WINDOW 0, 100, 0, 100
20 VIEWPORT 0, 100, 0, 100
```

orice figură geometrică va apare nedeformată pe orice display, deoarece pe ambele axe de coordonate se foloseşte aceeaşi unitate de măsură (cele două instrucţiuni realizează o corespondenţă unu la unu între unităţile utilizator – pe orizontală şi pe verticală şi unitatea grafică – UG).

TEST

Precizaţi care este semnificaţia următoarelor instrucţiuni:

```
10 VIEWPORT 75, 100, 75, 100
20 VIEWPORT 0, 50, 0, 50
```

Codificarea în limbajul BASIC–PRAE

"Căutaţi" originea axelor de coordonate în colţul din stînga sus al ecranului. Axa Ox v-o puteţi imagina ca pe o linie orizontală în partea de sus iar axa Oy ca pe o linie verticală în partea stîngă a ecranului. Ecranul TV este împărţit în 256 de puncte pe orizontală şi 256 de puncte pe verticală. Rezultă deci că valorile pot varia pe ambele axe de la 0 la 255.

Treziţi PRAE-ul! După cum constataţi, ecranul are culoarea albă. Dacă doriţi un fond negru al ecranului, tastaţi

SWITCH 1 : CLS

Dacă consideraţi că ecranul s-a . . . întristat, tastaţi

SWITCH 0 : CLS

şi ecranul va avea din nou fondul alb.

Observaţie. Pentru ştergerea ecranului folosiţi una din metodele: a) instrucţiunea **CLS** (Clear Screen); b) apăsarea simultană a tastei **[SHIFT]** şi **[S]**; instrucţiunea **PRINT CHR$** (0).

Pentru desenarea roboţelului pe ecranul televizorului, limbajul BASIC-PRAE vă pune la dispoziţie instrucţiunile grafice **PLOT (PLOTC)** şi **DRAW (DRAWC).** Să le examinăm pe rînd!

PLOT

Instrucţiunea **PLOT** serveşte la desenarea punctului (X, Y) pe ecran. Formatul general al instrucţiunii este:

Format general
PLOT ⟨X⟩, ⟨Y⟩

unde ⟨X⟩ şi ⟨Y⟩ reprezintă coordonatele punctului.

Remarci

- Punctul desenat cu **PLOT** are totdeauna culoarea opusă fondului ecranului.
- Instrucţiunea poate fi folosită şi în modul imediat.

Revenind la roboţel, nu trebuie decît să înlocuiţi pe **MOVE** (v. liniile 60, 85, 100) din programul BASIC-aMIC cu **PLOT.** Atenţie la spaţiul utilizator!

PLOTC

PLOTC (C de la clear) diferă de instrucţiunea de mai sus **(PLOT)** doar prin faptul că ea desenează cu puncte de culoarea ecranului.

Formatul general al instrucţiunii este

Format general
PLOTC ⟨X⟩, ⟨Y⟩

unde ⟨X⟩, ⟨Y⟩ reprezintă coordonatele punctului de pe ecran.

Observaţie. Instrucţiunea **PLOTC** poate fi folosită şi în modul imediat.

DRAW

Cea de-a treia instrucţiune grafică, **DRAW** are drept efect desenarea unui segment de dreaptă care uneşte ultimul punct desenat pe ecran cu punctul de coordonate (X, Y).

Formatul general al instrucţiunii este:

Format general
DRAW ⟨X⟩, ⟨Y⟩

Observaţii

- Pentru a fixa un capăt al segmentului de dreaptă se foloseşte (de obicei) instrucţiunea **PLOT.**
- Instrucţiunea **DRAW** poate fi folosită şi în modul imediat.

TEST

Desenaţi o diagonală a ecranului

```
10 PLOT 0, 0
20 DRAW 255, 255
```

Aplicaţie. Introduceţi şi executaţi următoarele programe BASIC–PRAE [ ]:

a)

```
10 PLOT 20, 20
20 DRAW 120, 20
30 DRAW 120, 120
40 DRAW 20, 120
50 DRAW 20, 20
```

b)

```
10 PLOT 0, 128
20 DRAW 255, 128
30 PLOT 128, 255
40 DRAW 128, 0
50 FOR I=0 TO 255
60 PLOT I, 128+10 * SIN
   (I * 6.28/255)
70 NEXT I
80 FOR J=0 TO 255
90 PLOTC J, 128+10 * SIN
   (I * 6.28/255)
100 NEXT J
110 SWITCH 1
120 CLS
```

DRAWC

DRAWC (C de la clear) diferă de instrucțiunea **DRAW** doar prin faptul că ea desenează cu puncte de culoarea ecranului.

Formatul general al instrucțiunii este:

Format general
DRAWC ⟨X⟩, ⟨Y⟩

unde ⟨X⟩, ⟨Y⟩ reprezintă coordonatele punctului de pe ecran.

Remarcă. Instrucțiunea **DRAWC** poate fi folosită și în modul imediat.

Acum, după ce ați aflat majoritatea secretelor privind utilizarea instrucțiunilor grafice **PLOT** și **DRAW** puteți trece fără emoții la desăvîrșirea roboțelului utilizînd limbajul BASIC–PRAE. Atenție la modul cum folosiți instrucțiunea **DRAW.** Nu uitați să înlocuiți și linia 105 (v. programul BASIC–aMIC).

□ Să desenăm un cilindru cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM

Vă mai amintiți cît de prietenos este AMSTRADUL! Ca să nu mai vorbim de HC-85, TIM S sau SPECTRUM. Este momentul să le trezim și pe ele. Să le punem să deseneze un . . . cilindru, de exemplu. Da, da chiar un cilindru a cărei generatoare este înclinată la 45° față de orizontală. Nici nu bănuiți cît de bine știu și ele să deseneze! Dar cum inteligența lor este funcție de . . . a noastră, nu ne mai rămîne decît să scriem programul care după cum veți vedea, nu este deloc complicat.

Ideea este următoarea. Se generează n cercuri de rază (r) constantă și se variază coordonatele centrului cercului (a, b) cu același pas (h), după cum urmează:

$$a_{k+1}=a_k+h \qquad (9)$$

$$b_{k+1}=b_k+h \qquad (10)$$

unde $n-1 \geqslant k \geqslant 1$, h avînd o valoare constantă.

Codificarea în limbajul BASIC–AMSTRAD

```
10 CLS
20 READ a, b, r
24 DATA 50, 50, 50
26 LOCATE 1,1 : PRINT "pasul este"; :
   INPUT h
27 CLS
28 n=50
30 FOR k=1 TO n
40   MOVE a+r, b
50     FOR i=0 TO 2 * PI STEP PI/60
60       DRAW a+r * COS(i),
         b+r * SIN(i)
70     NEXT i
72     a=a+h
75     b=b+h
80 NEXT k
90 GOTO 90
```

Programul inițializează coordonatele centrului primului cerc (v. liniile 20 și 24) fixînd pentru rază (r) valoarea 50. În ciclul **FOR–NEXT** (v. liniile

30–80) se desenează cilindrul propriu-zis după cum urmează: se trasează un cerc C(a,b,r) (v. liniile 50–70) după care se modifică valorile lui a şi b (v. liniile 72, 75) în conformitate cu relaţiile (9) şi (10). Bucla se repetă de 50 de ori (v. linia 28).

Este de datoria noastră să vă prezentăm mai întîi pe **LOCATE** (v. linia 26) şi după aceea să dăm bineţe unor cunoştinţe e drept mai vechi, întîlnite însă în alte limbaje, **MOVE** şi **DRAW**.

LOCATE

Instrucţiunea **LOCATE** plasează cursorul de text într-un anumit punct al ecranului ale cărui coordonate se raportează la colţul din stînga sus (1,1) al ecranului.

Observaţie. În **Mode** 1 există 40 de coloane şi 25 de linii, în **Mode** 0 nu există decît 20 de coloane dar tot 25 de linii, iar în **Mode** 2 sînt 80 de coloane şi 25 de linii.

Revenind la instrucţiunea **LOCATE** din program

```
26 LOCATE 1, 1 - PRINT "pasul este"; : INPUT h
```

veţi vedea în momentul execuţiei programului cum în colţul din stînga sus al ecranului de coordonate (1,1) în care prima cifră (1) reprezintă coordonata orizontală (cuprinsă între 1 şi 40 – v. **Mode** 1), iar cea de-a doua cifră (1) reprezintă coordonata verticală (cuprinsă între 1 şi 25) se afişează mesajul „pasul este" urmat de semnul întrebării (în aşteptarea introducerii dinamice a valorii pasului h).

Formatul general al instrucţiunii este:

Format general
LOCATE [#⟨număr canal⟩, ⟨X⟩, ⟨Y⟩

în care ⟨număr canal⟩ se consideră #∅, dacă nu îl precizaţi.

MOVE

Instrucţiunea **MOVE** poziţionează cursorul grafic în punctul de coordonate absolute specificat, fără a marca (fizic) punctul (pixel-ul) respectiv

```
40 MOVE a+r, b
```

Formatul general al instrucţiunii este:

Format general
MOVE ⟨X⟩, ⟨Y⟩ [,[⟨cerneală⟩] [,⟨tip cerneală⟩]

în care parametrul facultativ ⟨cerneală⟩ (cu valori cuprinse între 0 şi 15) permite schimbarea culorii stiloului grafic. Cît priveşte celălalt parametru opţional ⟨mod⟩ (cu valori între 0 şi 4–0 : normal, 1–**XOR**, 2–**AND**, 3–**OR**) el determină interacţiunea cernelii pe afişajul de pe ecran.

Remarci

- Cursorul grafic, diferit de cursorul caracter nu este vizibil.
- Coordonatele X şi Y sînt definite în raport cu colţul inferior din stînga ecranului de coordonate (0,0).
- Rezoluţia ecranului este de 640 (puncte orizontale)×400 (puncte verticale).
- Spre deosebire de instrucţiunea **LOCATE** aplicată caracterelor, aceste coordonate sînt aceleaşi în toate cele trei moduri de afişaj (**Mode** 0, 1 sau 2).
- Pentru poziţionarea cursorului grafic în puncte de coordonate relative se utilizează instrucţiunea **MOVER** (v. vol. 2, pag. 39).
- Pentru afişarea în mod grafic a unui punct de coordonate (X, Y) se utilizează instrucţiunea **PLOT** (v. vol. II pag. xxx). Spre deosebire de instrucţiunea **MOVE, PLOT** trasează pixel-ul (punctul) în coordonatele precizate.

DRAW

Instrucţiunea **DRAW** desenează o linie pe ecran între poziţia cursorului grafic şi o poziţie absolută specificată prin coordonatele X şi Y.

```
40 MOVE a+r, b
50 FOR i=0 TO 2 * PI STEP PI/60
60    DRAW a+r * COS(i), b+r * SIN(i)
70 NEXT i
```

Remarcă. Pentru trasarea cercului am generat arce mici de $\frac{PI}{60}$ radiani localizate prin puncte M_i de coordonate (a+r cos (i * α), b+r sin (iα)).
Segmentele care unesc aceste puncte aproximează cercul $\mathcal{C}$ (a, b, r), idee care a stat la baza desenării lui cu ajutorul instrucţiunilor **MOVE** şi **DRAW.**

Formatul general al instrucţiunii **DRAW** este

Format general
DRAW ⟨X⟩, ⟨Y⟩ [,[⟨cerneală⟩] [,⟨tip cerneală⟩]

Remarci

- Parametrii instrucţiunii au semnificaţia celor întîlniţi la instrucţiunea **MOVE.**
- Pentru trasarea unei drepte utilizînd coordonatele relative se foloseşte instrucţiunea **DRAWR** (v. vol. 2, pag. 21).

Acum, după ce aţi înţeles programul vă invităm să-l tastaţi şi apoi să-l executaţi. Aţi obţinut un cilindru a cărui generatoare este înclinată la 45° faţă de orizontală?

Aşa cum a fost scris, programul ciclează (v. linia 90) pînă cînd îl veţi întrerupe dvs. tastînd de două ori [**ESC**]. De notat că instrucţiunea

```
90 GO TO 90
```

evită întreruperea automată a programului după ultima linie şi afişarea mesajului **Ready.**

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

La HC-85, TIM S, SPECTRUM „căutaţi" punctul de coordonate (0, 0) în colţul din stînga jos al **ecranului** grafic (v. figura 12.17).

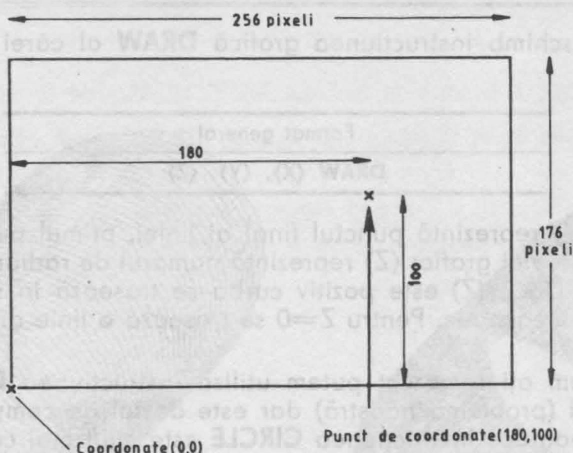


Fig. 12.17.

A nu se confunda cu punctul de coordonate (0,0) din colţul din stînga sus, al **ecranului alfanumeric** pe care l-am întîlnit în momentul utilizării instrucţiunii **PRINT AT.** Vă reamintim că **ecranul alfanumeric** conţine 24 de linii (liniile 22 şi 23 sînt rezervate mesajelor de sistem) a cîte 32 coloane (caractere) fiecare, caracterul de adresă alfanumerică (0,0) fiind în colţul din stînga sus al ecranului. HC-85, TIM S şi SPECTRUM permit însă şi o înaltă rezoluţie. Cum fiecare caracter alfanumeric este reprezentat pe o matrice de 8×8 pixeli rezultă că partea din ecran utilizabilă conţine (32×8)×(22×8) pixeli adresabili. Aşadar, pixelul de adresă (0,0) se află în colţul din stînga jos al **ecranului grafic.**

Imaginaţi-vă ecranul ca pe o casă cu mai multe nivele. Mergi la parter (X) şi apoi urci la etaj (Y) (v. reprezentarea punctului de coordonate 180 şi 100 din fig. 12.17).

Să ne reîntoarcem la problema noastră (trasarea unui cilindru a cărei generatoare este înclinată la 45° faţă de orizontală) pe care vrem s-o rezolvăm în BASIC HC-85, TIM S, SPECTRUM. Există mai multe posibilităţi. Fie cu instrucţiunile **PLOT** şi **DRAW** fie direct cu instrucţiunea **CIRCLE.**

În BASIC HC-85, TIM S, SPECTRUM instrucţiunea **PLOT** mută cursorul grafic din poziţia curentă la poziţia de coordonate (X,Y) şi marchează (aprinde) punctul (pixel-ul) respectiv.

Formatul general al instrucţiunii este

Format general
PLOTC ⟨X⟩, ⟨Y⟩

în care ⟨m⟩ reprezintă una din instrucţiunile **OVER** sau **INVERSE**; ⟨X⟩, ⟨Y⟩ reprezintă coordonatele punctului.

Observaţie. Instrucţiunea **PLOT INVERSE** 1; **OVER** 1; lasă pixel-ul neschimbat iar cursorul grafic rămîne poziţionat în punctul de coordonate (X, Y).

Dacă dorim să desenăm o linie lungă, devine plictisitor să folosim instrucţiunea **PLOT** (în cazul cînd dreapta se trasează punct cu punct). Pu-

tem folosi în schimb instrucţiunea grafică **DRAW** al cărei format general este

Format general
DRAW ⟨X⟩, ⟨Y⟩, ⟨Z⟩

în care ⟨X⟩, ⟨Y⟩ reprezintă punctul final al liniei, primul punct fiind definit de poziţia cursorului grafic; ⟨Z⟩ reprezintă numărul de radiani corespunzător circumferinţei. Dacă (Z) este pozitiv curba se trasează în sens invers mişcării acelor de ceasornic. Pentru Z=0 se trasează o linie dreaptă între cele două puncte.

După cum aţi remarcat putem utiliza instrucţiunea **DRAW** pentru a desena cercuri (problema noastră) dar este destul de complicat şi nu merită efortul, deoarece instrucţiunea **CIRCLE** este mult mai comodă şi, de ce să nu recunoaştem mai elegantă.

CIRCLE

Instrucţiunea **CIRCLE** trasează un cerc cu centrul în punctul (pixel-ul) de coordonate (x, y) de rază z pixeli. O vom utiliza în programul nostru.

```
10 CLS
20 READ a, b, r
24 DATA 50, 50, 50
26 PRINT AT 1, 1; "pasul este"; :
   INPUT h
27 CLS
28 LET n=50
30 FOR k=1 TO n
40 CIRCLE a, b, r
72 LET a=a+h
75 LET b=b+h
80 NEXT k
90 GOTO 90
```

Diferenţele de scriere a programului apar numai în liniile 26 şi 40. Cît priveşte cea de-a doua instrucţiune

```
40 CIRCLE a, b, r
```

ea înlocuieşte instrucţiunile

```
40  MOVE a+r, b
50  FOR i=0 TO 2 * PI STEP PI/60
60      DRAW a+r * COS (i), b+r * SIN (i)
70  NEXT i
```

din programul AMSTRAD.

Instrucţiunea **CIRCLE** din linia 40 trasează 50 de cercuri (v. fig. 12.18) de rază constantă (v. linia 20) avînd însă coordonatele centrului variabile (v. liniile 72 şi 75)

Formatul general al instrucţiunii este:

Format general
CIRCLE ⟨X⟩, ⟨Y⟩, ⟨Z⟩

în care, ⟨X⟩, ⟨Y⟩ reprezintă coordonatele centrului;
⟨R⟩ este raza cercului.

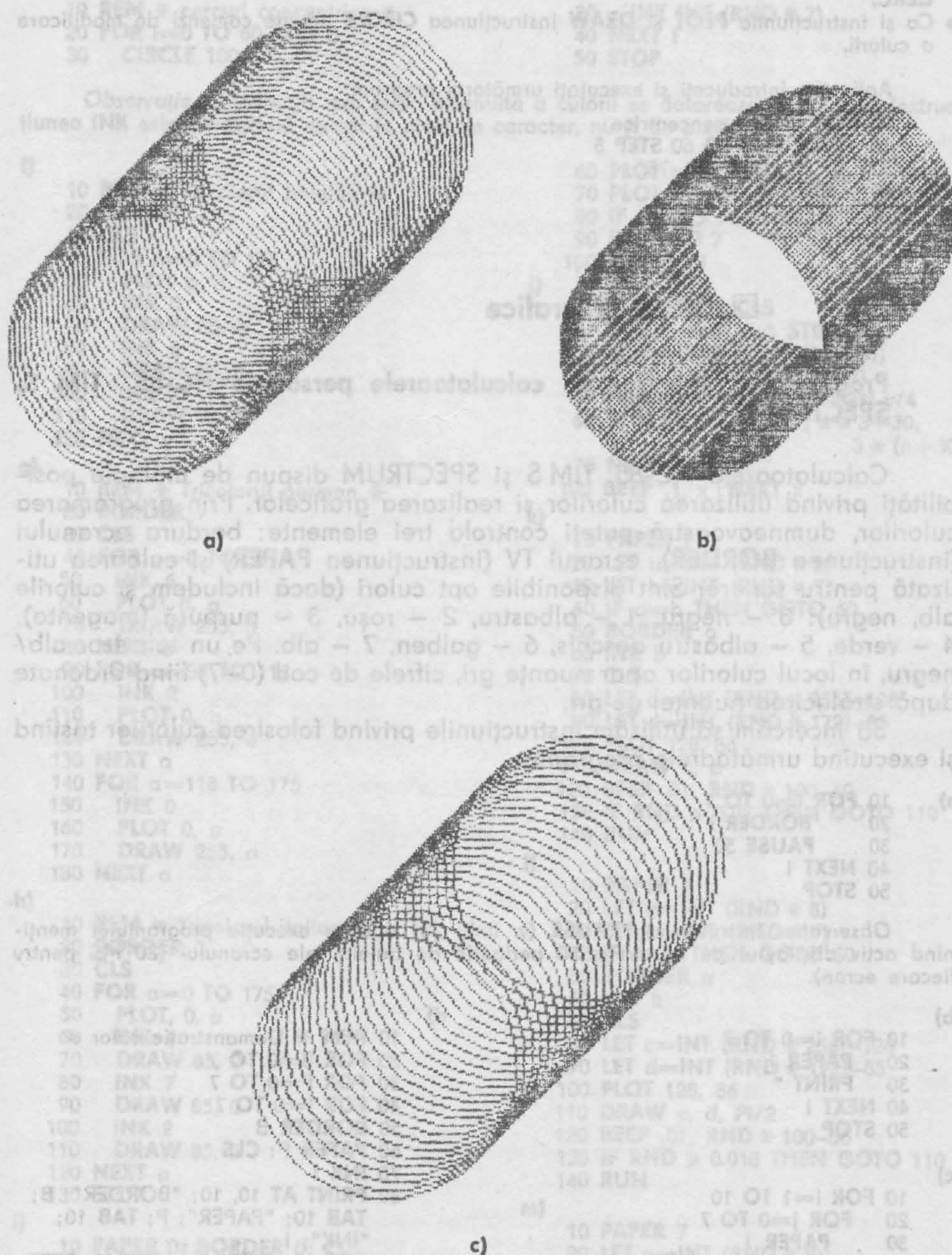


Fig. 12.18. Cilindru: *a)* pasul 2; *b)* pasul 1; *c)* pasul 3.

Remarci

- Instrucţiunea **CIRCLE** codifică în limbajul BASIC HC-85, TIM S, SPECTRUM primitiva **CERC.**
- Ca şi instrucţiunile **PLOT** şi **DRAW** instrucţiunea **CIRCLE** admite comenzi de modificare a culorii.

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 REM cercuri concentrice
20    FOR r=0 TO 60 STEP 5
30       CIRCLE 100, 75, r
40    NEXT r
50    STOP
```

□ Culori şi grafice

Programarea culorilor pe calculatoarele personale HC-85, TIM S, SPECTRUM

Calculatoarele HC-85, TIM S şi SPECTRUM dispun de multiple posibilităţi privind utilizarea culorilor şi realizarea graficelor. Prin programarea culorilor, dumneavoastră puteţi controla trei elemente: bordura ecranului (instrucţiunea **BORDER**), ecranul TV (instrucţiunea **PAPER**) şi culoarea utilizată pentru scriere. Sînt disponibile opt culori (dacă includem şi culorile alb, negru): 0 – negru, 1 – albastru, 2 – roşu, 3 – purpură (magenta), 4 – verde, 5 – albastru deschis, 6 – galben, 7 – alb. Pe un televizor alb/negru, în locul culorilor apar nuanţe gri, cifrele de cod (0–7) fiind ordonate după strălucirea nuanţei de gri.

Să încercăm să utilizăm instrucţiunile privind folosirea culorilor tastînd şi executînd următoarele programe:

a)
```
10 FOR i=0 TO 7
20    BORDER i
30    PAUSE 50
40 NEXT i
50 STOP
```

Observaţie. Instrucţiunea **PAUSE** (v. linia 30) opreşte execuţia programului menţinînd activ display-ul pe durata a 50 perioade de baleiaj ale ecranului (20 MS pentru fiecare ecran).

b)
```
10 FOR i=0 TO 7
20  PAPER i
30  PRINT "
40 NEXT i
50 STOP
```

c)
```
10 FOR i=1 TO 10
20   FOR j=0 TO 7
30     PAPER J
40     PRINT "   "
50   NEXT j
60 NEXT i
70 STOP
```

d)
```
10 REM * Demonstraţie color *
20 FOR B=0 TO 7
30 FOR P=0 TO 7
40 FOR I=0 TO 7
50 BORDER B
60 PAPER P: CLS
70 INK I
80 PRINT AT 10, 10; "BORDER"; B;
   TAB 10; "PAPER"; P; TAB 10;
   "INK"; I
90 FOR U=1 TO 60 : NEXT U : BEEP
   .15, P * I+2 * B
100 NEXT I
110 NEXT P
120 NEXT B
```

Observaţie. Instrucţiunea **BEEP** (v. linia 90) este utilizată pentru producerea sunetelor.

e)

```
10 REM * cercuri concentrice *
20 FOR i=0 TO 60 STEP 5
30    CIRCLE 100, 75, r
35    INK INT (RND * 7)
40 NEXT i
50 STOP
```

Observaţie. Distribuţia mai puţin obişnuită a culorii se datorează faptului că instrucţiunea **INK** este adresabilă numai la nivel de caracter, nu şi de pixel.

f)

```
10 REM * Tricolorul belgian *
20 BORDER 7
30 CLS
40 FOR a=0 TO 175
50    PLOT o, a
60    INK 0
70    DRAW 85, 0
80    INK 6
90    DRAW 85, 0
100   INK 2
110   DRAW 85, 0
120 NEXT a
```

g)

```
10 REM * Tricolorul german *
20 BORDER 7
30 CLS
40 FOR a=0 TO 58
50    INK 6
60    PLOT 0, a
70    DRAW 255, 0
80 NEXT a
90 FOR a=59 TO 117
100   INK 2
110   PLOT 0, a
120   DRAW 255, 0
130 NEXT a
140 FOR a=118 TO 175
150   INK 0
160   PLOT 0, a
170   DRAW 255, a
180 NEXT a
```

h)

```
10 REM * Tricolorul italian *
20 BORDER 0
30 CLS
40 FOR a=0 TO 175
50    PLOT, 0, a
60    INK 4
70    DRAW 85, 0
80    INK 7
90    DRAW 85, 0
100   INK 2
110   DRAW 85, 0
120 NEXT a
130 STOP
```

i)

```
10 PAPER 0: BORDER 0; CLS
20 LET C=255 : LET d=175
30 INK RND * 7
40 LET a=c * RND
50 LET b=d * RND
60 PLOT a, b : PLOT a, d-b
70 PLOT c-a, b: PLOT c-a, d-b
80 IF RND >. 5 THEN GOTO 60
90 INK RND 7
100 GOTO 50
```

j)

```
10 BORDER 2 : CLS
20 FOR x=0 TO 63 STEP .5
30 LET y=20 * SIN (x/32 * PI)
40 IF y=0 THEN GOTO 100
50 FOR n=0 TO y STEP SGN y/4
60 PLOT INK RND * 6; x * 3+30,
                        3 * (n+30)
70 NEXT n
100 BEEP .1, x : NEXT x
```

k)

```
10 PAPER 7
20 LET a=INT (RND * 8)
30 LET b=INT (RND * 7)
40 IF a=b THEN GOTO 60
50 BORDER a
60 INK b
70 CLS
80 LET c=INT (RND * 256)-128
90 LET d=INT (RND * 172)-85
100 PLOT 128, 86
110 DRAW c, d
120 BEEP .01, RND * 100-50
130 IF RND > 0.02 THEN GOTO 110
140 RUN
```

l)

```
10 PAPER 7
20 LET a=INT (RND * 8)
30 LET b=INT (RND * 7)
40 IF a=b THEN GOTO 60
50 BORDER a
60 INK b
70 CLS
80 LET c=INT (RND * 256)-128
90 LET d=INT (RND * 172)-85
100 PLOT 128, 86
110 DRAW c, d, PI/2
120 BEEP .01, RND * 100-50
130 IF RND > 0.015 THEN GOTO 110
140 RUN
```

m)

```
10 PAPER 7
20 LET a=INT (RND * 8)
30 LET b=INT (RND * 7)
40 IF a=b THEN GOTO 60
50 BORDER a
60 INK b
```

```
70 CLS
80 LET c=INT (RND * 256)-128
90 LET d=INT (RND * 172)-85
100 PLOT 128, 86
110 IF RND > 0.5 THEN INK RND * 6
120 DRAW c, d, PI/2
130 BEEP .01, RND * 100-50
140 IF RND > 0.015 THEN GOTO 110
150 RUN
```

n)

```
10 BORDER 5: PAPER 7: CLS
20 CIRCLE INK RND * 6;
   128+RND * 10-RND * 10,
   85+RND * 7-RND * 7, RND * 65
30 IF RND > . 92 THEN CLS
40 BEEP RND/3, RND * 100-30
50 GOTO 20
```

o)

```
10 PAPER 7: CLS
20 INK 0
30 BORDER 7
40 FOR a=1 TO 10
50 FOR J=1 TO 7 STEP .1
60 PLOT a * J * COS (J)+110,
   a * J * SIN(J)+100
70 NEXT J
80 INK a/2
90 NEXT a
```

p)

```
10 PAPER 7: CLS
20 INK 0
30 BORDER 7
40 FOR a=5 TO 19
50 FOR J=1 TO 7 STEP .3
60 PLOT a * J * COS (J)/2+110,
   a * J * SIN (J)+100
70 NEXT J
80 NEXT a
```

r)

```
10 PAPER 7: CLS
20 INK 0
30 BORDER 7
40 FOR a=2 TO 19
50 FOR J=1 TO 7 STEP .01
60 PLOT a * (J+4 * SIN (J)) *
   COS (J)+110, a * (J+2 * SIN
   (J) * SIN (J)+100
70 NEXT J
80 INK a/3.5
90 NEXT a
```

Programarea culorilor pe AMSTRAD

În BASIC-AMSTRAD aveţi de ales între 27 de culori! În **Mode** 0 puteţi avea în acelaşi timp pe ecran 16 culori, în **Mode** 1–4 culori iar în **Mode** 2 numai 2 culori. Pentru programarea culorilor utilizaţi instrucţiunile **BORDER, PAPER, PEN, INK** al căror format general este prezentat în volumul 2, pag. 14, 44, 30.

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 MODE 1 : INK 2, 10 : INK 3, 4
20 ORIGIN 440, 100, 440, 640, 100, 300
30 WINDOW 1, 26, 1, 25
40 CLG 2 : GRAPHICS PAPER 0
50 DRAW 200, 200, 3
60 MOVE 2,0 : FILL 3
70 ORIGIN 440, 0, 440, 640, 0, 400
80 GRAPHICS PEN, 1 : MASK, 0
90 FOR y=60 TO 318 STEP 2
100   GOSUB 220
110   FRAME : FRAME
120   GOSUB 220
130 NEXT
140 TAG
150 FOR y=60 TO 318 STEP 2
160   MOVE 96, y : PRINT CHR$ (224) ;
170   FRAME : FRAME
180   MOVE 96, y : PRINT CHR$ (224) ;
190 NEXT
200 END
210 MOVE 90, y, 1
220 DRAWR 20, 0 , , 1
230 DRAWR 0,20
240 DRAWR -20, 0
250 DRAWR 0, -20
260 RETURN
```

Observaţie. Pentru înţelegerea instrucţiunilor folosite de acest program consultaţi volumul 2, paginile 43, 28, 58.

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)

```
20 ON BREAK GOSUB 220
30 FOR i=1 TO 15 : INK i, 26 : NEXT
40 m(1)=1 : m(2)=2 : m(3)=4 : m(4)=8
50 MODE 0 : PRINT CHR$(23); CHR$(3); : TAG
```

```
60 FOR p=1 TO 4
70 GRAPHICS PEN m(p), 1
80 LOCATE #1, 1, 25 : PRINT #1, CHR$(48+p);
90 FOR x=0 TO 7
100 FOR y=0 TO 14 STEP 2
110 IF TEST (x * 4, y)=0 THEN 140
120 MOVE (x+6) * 32, (y+6) * 16 : PRINT CHR$(143);
130 MOVE (x+6) * 32, (y+7) * 16 : PRINT CHR$(143);
140 NEXT y, x, p
150 LOCATE #1, 1, 25 : PRINT #1, " ";
160 FOR p=1 TO 4
170 FOR i=1 TO 25 : FRAME : NEXT
180 FOR i=0 TO 15
190 IF (i AND m(p))=0 THEN INK i, 0 ELSE INK i, 26
200 NEXT i, p
210 GOTO 160
220 INK 1, 26
RUN
```

b)

```
20 DEFINT a-z
30 INK 0,1 : INK 1,26
40 INK 2,6 : INK 3,6
50 FOR i=4 TO 7 : INK i, 9 : NEXT
60 FOR i=8 TO 15 : INK i, 20; NEXT
70 MODE 0 : DEG : ORIGIN 0,150 : CLG : MOVE 0,150
80 FOR x=16 TO 640 STEP 16
90 DRAW x, COS (x) * 150+RND * 100,4
100 NEXT
110 MOVE 0,0 : FILL 4
120 cx=175 : GOSUB 320
130 cx=525 : GOSUB 320
140 SYMBOL 252, 0, 0, &C, &1F, &30, &7F, &FF
150 SYMBOL 253, 0, 6, &E, &F2, 2, &F2, &FF
160 SYMBOL 254, 0, &60, &70, &7F, &7F, &7F, &7F
170 SYMBOL 255, 0, 0, 0, &F8, &EC, &FE, &FF
180 pr$=CHR$(254)+CHR$(255)
190 pl$=CHR$(252)+CHR$(253)
200 TAG : t ! = TIME
210 FOR x=-32 TO 640 STEP 4
220 x2=((608-x) * 2) MOD 640 : h1=RND * 10 : hr=50 * SIN(x)
230 GRAPHICS PEN 8, 1 : MOVE x, 100+hr , , 3 : PRINT pr$ ;
240 GRAPHICS PEN 2, 1 : MOVE x2, 115+h1 , , 3 : PRINT pl$ ;
250 IF (TEST (x2-2, 115+h1-12) AND 8)=8 THEN 380
260 IF TIME-t ! < 30 THEN 260
270 FRAME : t ! = TIME
280 GRAPHICS PEN 7, 1 : MOVE x, 100+hr , , 2 : PRINT pr$ ;
290 GRAPHICS PEN 13, 1 : MOVE x2, 115+h1 , , 2 : PRINT pl$ ;
300 NEXT
310 GOTO 210
320 MOVE cx, 100
330 FOR x=0 TO 360 STEP 10
340 DRAW cx+SIN(x) * 50+10 * RND, 100+COS(x) * 25+10 * RND, 1
350 NEXT
360 DRAW cx, 100 : MOVE cx, 90 : FILL 1
370 RETURN
380 ENT -1, 1, 1, 1
390 SOUND 1, 25, 400, 15 , , 1, 15
400 FOR y=100+hr TO -132 STEP -2
410 GRAPHICS PEN 7, 1 : MOVE x, y , , 2 : PRINT pr$ ;
420 GRAPHICS PEN 8, 1 : MOVE x, y-2 , , 3 : PRINT pr$ ;
430 NEXT
440 GOTO 70
RUN
```

□ Generarea unui rezervor-display cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM. Trasarea histogramelor cu ajutorul caracterelor mozaic

Să nu credeţi că am uitat de problema rezervoarelor cu benzină care ne-a însoţit pe tot parcursul lucrării. Nu poate fi vorba de aşa ceva. Am abandonat-o numai pentru o clipă spre a vă putea prezenta facilităţile grafice de care dispun calculatoarele cu care am lucrat. O întrebare. Vă mai amintiţi de listele cu livrările de benzină, generate în conversaţiile precedente? Din păcate acestea nu sînt de mare ajutor atunci cînd este vorba de interpretarea rezultatelor. În foarte multe cazuri sîntem interesaţi de modul cum se schimbă valorile şi nu de precizia obţinerii lor.

O metodă eficientă, pe care v-o propunem în această conversaţie, constă în vizualizarea rezultatelor folosind un *grafic* sau o *histogramă*. Astfel de imagini sînt mult mai sugestive chiar dacă de multe ori precizia are de suferit.

Să începem prin a genera din caractere mozaic un *rezervor-display* căruia să-i alăturăm şapte histograme reprezentînd cantităţile de benzină livrate în fiecare zi a săptămînii din acel rezervor.

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 262

Caractere mozaic

Pentru a „produce" *rezervorul-display* vom utiliza simbolurile grafice mozaic ale calculatorului AMSTRAD, după cum urmează:

Baza *rezervorului-display* o vom construi prin tipărirea a trei caractere mozaic avînd codurile 130, 131 şi 129. Pereţii îi vom realiza prin tipărirea a două caractere mozaic cu codurile 138 (pentru peretele din stînga) şi 133 (pentru peretele din dreapta). În sfîrşit, capacul *rezervorului-display* îl vom „pune" cu codurile 136, 140 şi 132.

Putem reprezenta cantitatea totală de combustibil livrată prin desenarea unei linii verticale a cărei lungime este proporţională cu valoarea livrărilor efectuate în cursul unei săptămîni. Înălţimea *rezervorului-display* va fi suficientă pentru a permite coloanei să atingă valoarea maximă. În vederea citirii cantităţilor livrate vom desena o scară chiar lîngă linia măsurătoare. Cît priveşte desenarea histogramelor (pentru cele şapte zile) vom utiliza, de asemenea, caractere semigrafice (pline şi semipline).

Specificaţiile de programare şi documentaţia de proiectare sînt ilustrate în modulele de analiză şi proiectare structurată, figura 12.19.

Acum, este momentul să vă prezentăm lista programului (v. vol. 2. pag. 262), pe care vă invităm să-l tastaţi şi după aceea să-l şi executaţi.

După ce în linia 140 s-a iniţializat abscisa de unde va începe trasarea *rezervorului-display*, în liniile următoare (150–170) se generează *rezervorul-display* propriu-zis. Scalarea *rezervorului-display* se realizează cu secvenţa de instrucţiuni 175–190, iar afişarea iniţialelor (L, M, . . . , D) zilele săptămînii se face cu instrucţiunile 192–196. Vom introduce dinamic (v. li-

TABELA DE VARIABILE		
Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC-85, TIM S, SPECTRUM)		
Variabile de intrare	Variabile de stare	Variabile de ieşire
t: cantitatea (totală) de combustibil livrată	z, i, n, j, y: variabile de lucru t: cantitatea de combustibil livrată zilnic	t: cantitatea de combustibil livrată

a)

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC–85, TIM S, SPECTRUM)

Descrierea programului

Programul desenează cu ajutorul caracterelor mozaic un rezervor-display şi o familie de histograme ce reprezintă cantitatea de combustibil (benzină) livrată în cursul unei săptămini (defalcată pe fiecare zi în parte).

Intrări

Se introduce de la tastatură cantitatea de combustibil livrată intr-o săptămînă.

Ieşiri

Un rezervor-display şi şapte histograme.

Lista de funcţiuni ale programului

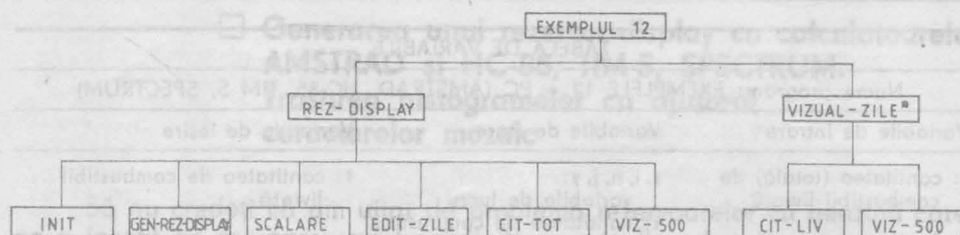
1. Iniţializare abscisă generare rezervor-display.
2. Reprezentare bază.
3. Reprezentare corp.
4. Reprezentare capac.
5. Scalare rezervor-display.
6. Afişare iniţiale zile săptămînă.
7. Citire cantitate totală de combustibil livrată.
8. Citire cantitate de combustibil livrată zilnic.
9. Calcul număr întreg de unităţi de măsură prin care se reprezintă o cantitate t de benzină livrată.
10. Reprezentare printr-un număr de Y/2 caractere "pline" (CHR$ (143))
11. Reprezentare printr-un număr de (y–1)/2 caractere "pline" plus un caracter "semiplin" (CHR$(140)).

b)

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE	
Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC–85, TIM S, SPECTRUM)	
Modul	**Funcţiuni**
INIT	1
GEN–REZ–DISPLAY	2, 3, 4
SCALARE	5
EDIT–ZILE	6
CIT–TOT	7
CIT–LIV	8
VIZ–500	9, 10, 11

c)

Fig. 12.19. Modulele de analiză şi proiectare structurată: *a)* tabela de variabile a programelor BASIC-AMSTRAD şi BASIC HC-85, TIM S, SPECTRUM; *b)* specificaţii de programare ale programelor BASIC-AMSTRAD şi BASIC HC-85, TIM S, SPECTRUM; *c)* alocarea funcţiunilor de prelucrare;



d)

(0,0) ox

CHR$(136) CHR$(140) CHR$(132)

100
90
80
70
60
50
40
30
20
10
0

CHR$(138) CHR$(133)

CHR$(130) CHR$(131) CHR$(129)

oy

e)

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```
MOZAMS   SEQ
         CLS
         Z=8
         I=0
         LOCATE Z, 20
         PRINT CHR$(130); CHR$(131); CHR$(129)
```

Fig. 12.19 f)

Fig. 12.19. *d)* diagrama de structură a programelor BASIC-AMSTRAD și BASIC HC-85, TIM S, SPECTRUM; *e)* rezervorul-display AMSTRAD; *f)* pseudocodul programului BASIC-AMSTRAD.

```
                    PSEUDOCODUL
         Nume program: EXEMPLUL 12 – PC (AMSTRAD)

A         PENTRU N DE LA 1 LA 11
             LOCATE Z–1, 20–N
             PRINT "–" CHR$(138); CHR$(128); CHR$(133)
A         SFÎRŞIT
          LOCATE Z, 8
          PRINT CHR$(136); CHR$(140); CHR$(132)
B         PENTRU N DE LA 1 LA 11
             LOCATE (Z–6), 20–N
             PRINT I
             I=I+10
B         SFÎRŞIT
C         PENTRU J DE LA Z+1 LA 38
             LOCATE J, 20
             PRINT CHR$(131)
C         SFÎRŞIT
          LOCATE 15, 22
          PRINT "L M M J V S D"
          LOCATE 1, 1
          PRINT "Cantitatea de combustibil rezervor"
          INPUT T
          LOCATE 1, 1
          PRINT "                                  "
          DO 500
D         PENTRU Z DE LA 14 LA 32 PAS 3
             READ T
             DO 500
D         SFÎRŞIT
MOZAMS    END
VIZ–500   SEQ
A         PENTRU N DE LA 1 LA 11
             LOCATE Z+1, 20–N
             PRINT CHR$(128)
A         SFÎRŞIT
          Y=INT((T+5)/5+0.5)
B         PENTRU N DE LA 1 LA INT (Y/2)
             LOCATE Z+1, 20–N
             PRINT CHR$(143)
B         SFÎRŞIT
C         DACĂ INT (Y/2) < > Y/2
             Y=INT (Y/2)
             LOCATE Z+1, 19–Y
             PRINT CHR$(140)
C         SFÎRŞIT
VIZ–500   END
```

Fig. 12.19.f)

niile 205–206) cantitatea totală de combustibil livrată în cursul unei săptămîni după care se vizualizează cu subrutina 500. Cele şapte histograme reprezentînd livrările zilnice de combustibil se tipăresc cu instrucţiunile 210–215.

Să nu părăsim programul înainte de-a înţelege bine modul în care a fost realizată subrutina de vizualizare livrări (500).

Histograme

În instrucţiunea 530 cantitatea de combustibil (benzină) ce urmează a fi vizualizată se consideră t+5 avînd în vedere că originea (punctul 0) se află la o „înălţime" de 5 unităţi de măsură faţă de baza *rezervorului-display* (înălţimea unui caracter se consideră egală cu 10 unităţi de măsură – v. figura 12.19.e). Cantitatea Y=**INT** ((t+5)/5+0 . 5) semnifică numărul întreg de cîte 5 unităţi de măsură prin care se reprezintă o cantitate de combustibil t.

Exemplele următoare de conversie (v. tabelul 12.2) în jumătăţi de caracter (5 unităţi de măsură) vă vor ajuta să înţelegeţi mai bine mecanismul de reprezentare.

Tabelul 12.2

t	y
0	**y=int((0+5)/5+0.5)=int(1.+0.5)=int(1.5)=1**
2	**y=int((7)/5+0.5)=int(1,4+0,5)=int(1,1)=1**
3	**y=int(8/5+0,5)=int(1,6+0,5)= int(2,1)=2**
5	**y=int(10/5+0,5)=int(2+0,5)=int(2,5)=2**
7	**y=int(12/5+0,5)=int(2,4+0,5)=int(2,9)=2**
8	**y=int(13/5+0,5)=int(2,6+0,5)=int(3,1)=3**
12	**y=int(17/5+0,5)=int(3,4+0,5)=int(3,9)=3**

Remarcaţi o rotunjire a valorii cantităţilor de combustibil livrate în jumătăţi de caracter, după regula: **o cantitate t se converteşte în acelaşi număr de jumătăţi de caracter ca şi cea mai apropiată cantitate divizibilă cu 5.**

Astfel, cantităţile t=73, t=74, t=75, t=76, t=77 vor avea aceeaşi reprezentare grafică. Analog, cantităţile t=78, t=79, t=80, t=81, t=82. Rezultă, după cum aţi putut constata o anumită imprecizie. Deoarece în reprezentarea cantităţilor nu putem folosi decît caracterul „plin" – **CHR$**(143) sau caracterul „semiplin" – **CHR$**(140) apare ca o necesitate utilizarea instrucţiunilor 530–565. Ele au următoarea semnificaţie: în cazul cînd numărul Y rezultat prin conversia lui t este par, reprezentarea acestei cantităţi se va face printr-un număr de Y/2 caractere „pline" – **CHR$**(143), în caz contrar reprezentarea făcîndu-se printr-un număr de (Y–1)/2 caractere pline plus un caracter semiplin.

TEST

Modificaţi programul anterior astfel încît să se afişeze (cu ajutorul caracterelor mozaic) cantităţile de combustibil (defalcate pe zile) din trei rezervoare. Cele trei cantităţi se vor introduce de la tastatură.

```
200 FOR d=1 TO 3
205 LOCATE 1,1 : PRINT "Cantitatea de combustibil rezervor"; d
206 INPUT t
.
.
.
215 NEXT Z
300 NEXT d
```

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 MODE 1
20 x=20 : y=12
22 c=249
30 LOCATE x, y : PRINT CHR$ (c)
40 a$=INKEY$
45 IF ASC (a$)<240 OR ASC (a$)>243 THEN 40
50 IF a$=CHR$(240) THEN yadd=-1 : xadd=0 : c=248
60 IF a$=CHR$(241) THEN yadd=+1 : xadd=0 : c=249
70 IF a$=CHR$(242) THEN xadd=-1 : yadd=0 : c=251
80 IF a$=CHR$(243) THEN xadd=+1 : yadd=0 : c=250
85 q1dx=x : q1dy=y
90 x=x+xadd : y=y+yadd
100 IF x>40 THEN x=40
110 IF x<1 THEN x=1
120 IF y>25 THEN y=25
130 IF y<1 THEN y=1
135 LOCATE q1dx, q1dy : PRINT " "
140 GO TO 30
```

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

Caracterele mozaic cu ajutorul cărora puteţi construi atît *rezervorul-display* cît şi cele şapte histograme sînt ilustrate în figura 12.20.

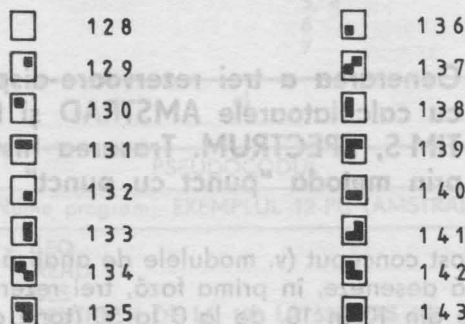


Fig. 12.20.

Aplicaţie. Introduceţi şi executaţi următorul program. Comparaţi rezultatele cu fig. 12.20.

```
10 CLS
20 PRINT
30 FOR i=32 TO 164
40    PRINT CHR$ i ; " " ;
50 NEXT i
60 STOP
```

Odată cunoscute: setul caracterelor mozaic, rezoluţia ecranului, instrucţiunile semigrafice nu vă rămîne decît să adaptaţi programul de pe AMSTRAD pe calculatoarele HC-85, TIM S, SPECTRUM. Luaţi această invitaţie ca pe o provocare la programare!

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)

```
10 DIM a(7)
20 DIM c(7)
30 FOR s=1 TO 30
40 CLS
50 FOR p=1 TO 7
60   LET a(p)=122+INT
       (RND * 22)
70   LET c(p)=INT (7 * RND)
80 NEXT p
90 LET k=3+INT (5 * RND)
100 FOR n=1 TO 672 STEP k
110   FOR j=1 TO k
120   PRINT INK c(j); CHR$ a(j)
130   NEXT j
140 NEXT n
150 PAUSE 200
160 NEXT s
```

b)

```
10 DIM a(4,4)
20 FOR j=1 TO 4
30   FOR i=1 TO 4
40     READ a(i, j)
50   NEXT i
60 NEXT j
70 DATA 128, 133, 143, 128
80 DATA 132, 140, 142, 140
90 DATA 128, 132, 142, 128
100 DATA 128, 138, 128, 138
110 LET r=10
120 LET c=15
130 FOR j=1 TO 4
140   FOR i=1 TO 4
150     PRINT AT r+j-1, c+i-1;
        CHR$ a(i, j);
160   NEXT i
170 NEXT j
180 STOP
```

c)

```
10 DIM a(4,4)
20 FOR j=1 TO 4
30   FOR i=1 TO 4
40     READ a(i, j)
50   NEXT i
60 NEXT j
70 DATA 128, 133, 143, 128
80 DATA 132, 140, 142, 140
90 DATA 128, 132, 142, 128
100 DATA 128, 138, 128, 138
110 FOR r=0 TO 16 STEP 5
120   FOR c=0 TO 28 STEP 4
130     FOR j=1 TO 4
140     FOR i=1 TO 4
150       PRINT AT r+j-1, c+i-1:
          CHR$ a(i, j);
160     NEXT i
170     NEXT j
180   NEXT c
190 NEXT r
200 STOP
```

□ Generarea a trei rezervoare-display cu calculatoarele AMSTRAD şi HC-85, TIM S, SPECTRUM. Trasarea histogramelor prin metoda "punct cu punct"

Programul a fost conceput (v. modulele de analiză şi proiectare structurată, fig. 12.21) să deseneze, în prima fază, trei rezervoare-display (v. figura 12.22) gradate din 10 în 10, de la 0 la 50 (tone combustibil). În cea de-a doua fază am programat conversaţia propriu-zisă. După ce comunicaţi

TABELA DE VARIABILE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC–85, TIM S, SPECTRUM)

Variabile de intrare	Variabile de stare		Variabile de ieşire
t: cantitatea de combustibil (în tone) din rezervor	d:	număr rezervor	t: cantitatea de combustibil (în tone) din rezervor
	x, y:	coordonate poziţionare cursor	
	i, k, q, r, y, x, p, j, q, l:	variabile de lucru	

a)

Fig. 12.21. *a)* tabela de variabile;

SPECIFICAŢII DE PROGRAMARE

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Descrierea programului

Programul desenează pe ecranul monitorului trei rezervoare-display. Fiecare rezervor-display ilustrează cantitatea de combustibil (în tone) livrată la un moment dat. Modul de construire a imaginilor este "punct cu punct".

Intrări

Se introduc de la tastatură cantităţile (în tone) de combustibil livrate (pentru trei rezervoare).

Ieşiri

Trei rezervoare-display.

Lista de funcţiuni ale programului

1. Iniţializare x, y
2. Curăţire ecran
3. Poziţionare cursor grafic
4. Trasare cadru rezervor-display
5. Trasare bază rezervor-display
6. Gradare rezervor-display
7. Citire dinamică valoare cantitate combustibil rezervor
8. Reprezentare parte activă rezervor-display

b)

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC–85, TIM S, SPECTRUM)

Modul	**Funcţiuni**
INIT	1, 2
CADRU–HIST	3, 4
RA–340	5, 8
GRAD–HIST	6
CIT–CANT	7

c)

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (AMSTRAD)

```
PUNCT   SEQ
        READ Y
        CLS
A       PENTRU X DE LA 100 LA 630 PAS 200
          MOVE X, Y
          DRAWR 70, 0
          DRAWR 0, 300
          DRAWR -70, 0
          DRAWR 0, -300
          Q=50
          K=0
          DO 340
A1        PENTRU I DE LA Y+50 LA Y+250 PAS 8
            MOVE X, I
            R=(I-Y-50)/40
```

d)

Fig. 12.21. *b)* specificaţii de programare; *c)* alocarea funcţiunilor de prelucrare; *d)* pseudocodul pentru programul BASIC-AMSTRAD;

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```
A2        DACĂ INT (R)=R
            DRAWR -20, 0
            LOCATE (X-70)/16, (400-I)/16
            PRINT K
            K=K+10
A2        IN CAZ CONTRAR
            DRAWR -10, 0
A2        SFÎRŞIT
A1       SFÎRŞIT
A       SFÎRŞIT
        Q1=30
B       PENTRU X DE LA 100 LA 630 PAS 200
        D=D+1
        LOCATE 1, 1
        PRINT "CANTITATEA ÎN TONE REZERVOR" D
        INPUT T
        P=50
        Q=T*4
        DO 340
        LOCATE 1, 1
        PRINT "                              "
B       SFÎRŞIT
PUNCT   END
RA-340  SEQ
        MOVE X, Y+P
A       PENTRU I DE LA Y+P LA Y+P-1+Q
A1       PENTRU J DE LA X-1+Q1 LA X+70-Q1
           DRAWR 0, 1
           MOVE J+1, I
A1       SFÎRŞIT
A       SFÎRŞIT
        DRAWR 0, 1
RA-340  END
```

d)

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (HC-85, TIM S, SPECTRUM)

```
HCPUNCT   SEQ
          READ Y
          CLS
          P=0
          Q1=0
A         PENTRU X DE LA 40 LA 200 PAS 80
            PLOT X, Y
            DRAW 30,0
            DRAW 0, 130
            DRAW -30, 0
            DRAW 0, -130
            Q=20
            K=0
            Q1=28
            DO 340
```

e)

Fig. 12.21. *e)* pseudocodul pentru programele BASIC HC-85, TIM S, SPECTRUM;

PSEUDOCODUL	
	Nume program: EXEMPLUL 12 – PC (HC-85, TIM S, SPECTRUM)

```
A1          PENTRU I DE LA Y+20 LA Y+120 PAS 4
              PLOT X, I
              R=(I–Y–20)/20
A2            DACĂ INT(R)=R
                DRAW –20, 0
                PRINT AT (175–I)/8, (X–30)/8; K
                K=K+10
A2            IN CAZ CONTRAR
                DRAW –10, 0
A2            SFÎRŞIT
A1          SFÎRŞIT
A         SFÎRŞIT
            Q1=2
            D=0
B           PENTRU X DE LA 40 LA 200 PAS 80
              D=D+1
              PRINT AT 1, 1; "cantitatea in tone rezervor"; d
              INPUT t
              P=20
              Q=T * 2
              DO 340
              PRINT AT 1, 1; "                    "
B           SFÎRŞIT
HCPUNCT     END
340         SEQ
              PLOT X, Y+P
A             PENTRU I DE LA Y+P LA Y+P–1+Q
A1              PENTRU J DE LA X+(30–Q1)/2 LA X+(30–Q1)/2+Q1
                  DRAW 0, 1
                  PLOT J, I
A1              SFÎRŞIT
A             SFÎRŞIT
            DRAW 0, 1
340         END
```

e)

Fig. 12.21. *f)* diagrama de structură.

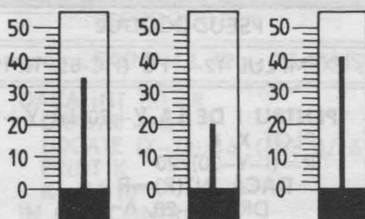


Fig. 12.22. Rezervoare-display generate prin metoda punct cu punct.

calculatorului, pentru fiecare rezervor în parte, cantităţile de combustibil livrate puteţi admira pe ecranul monitorului histogramele corespunzătoare, trasate prin metoda „punct cu punct".

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 263

Lista programului se află în vol. 2, pag. 263. Înainte de a-l tasta, vă sugerăm să-l analizaţi cu multă atenţie.

40–240 Multiplică pe ecranul monitorului trei *rezervoare-display* gradate de la 0–50 (tone combustibil)

50– 90 Desenează cadrul *rezervorului-display*

120 Trimite la subrutina 340 care pentru q=50, p=0, q1=0 desenează baza *rezervorului-display*

130–230 Gradează *rezervorul-display*. Gradarea se face din 5 în 5. Numărul de tone este vizualizat din 10 în 10 (v. linia 190); instrucţiunea 170 utilizează formula de trecere de la sistemul de coordonate pentru **DRAW** la sistemul de coordonate pentru **LOCATE.**

260–320 Desenează „partea activă" a *rezervorului-display* corespunzătoare celor trei cantităţi de combustibil livrate ale căror valori se introduc de la tastatură (v. linia 280).

340–420 Subrutina construieşte „punct cu punct" cu ajutorul a două bucle **FOR-NEXT** (350–400) şi (360–390) un dreptunghi de lungime.
L=x+70–q1–x–1–q1=69–2q1 puncte, şi de lăţime
l=y+p–1+q–y–p=q–1 puncte.
Un punct este desenat cu ajutorul instrucţiunilor 370, 380. Variabilele q şi q1 reglează lăţimea şi lungimea acestui dreptunghi, fapt ce permite atît desenarea bazei *rezervorului-display* cît şi reprezentarea „părţii active" a acestuia.

Observaţie. Datorită modului de lucru "punct cu punct" viteza de reprezentare este mică.

TEST

Simulaţi cu creionul şi hîrtia modul de funcţionare a subrutinei 340 de trasare "punct cu punct" a părţii active a rezervorului-display (v. liniile 290, 300, 310 din programul principal).

```
340 MOVE x, y+p
350 FOR i=y+p TO y+p-1+q
```

```
360  FOR j=x-1+q1 TO x+70-q1
370     DRAWR 0,1
380     MOVE J+1, i
390  NEXT j
400  NEXT i
410  DRAWR 0,1
420  RETURN
```

Aplicaţie. Introduceţi şi executaţi următorul program:

```
10 MODE 0 : BORDER 13
20 MOVE 0, 200 : DRAW 640, 200
30 FOR x=80 TO 560 STEP 80
40 MOVE x, 0 : DRAW x, 400
50 NEXT : MOVE - 40, 300
60 FOR C=0 TO 7
70 MOVER 80, 0 : FILL c
80 MOVER 0, - 200 : FILL c+8
90 MOVER 0, 200 : NEXT
100 GO TO 100
```

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 266

Pentru scrierea programului am avut în vedere atît rezoluţia ecranului cît şi setul de instrucţiuni grafice disponibile pe HC-85, TIM S şi SPECTRUM. Am procedat prin a revedea specificaţiile de programare ale exemplului precedent după care am întocmit o nouă documentaţie de proiectare (v. pseudocodul din fig. 12.21.e). Lista programului este prezentată în vol. 2, pag. 266. Vă rugăm să lecturaţi în paralel cele două programe, ca apoi, singuri, fără ajutorul nostru să descoperiţi . . . diferenţele. Atenţie la parametrii care se transmit subrutinei 340!

TEST

În urma execuţiei uneia din versiunile programului BASIC HC-85, TIM S, SPECTRUM de mai sus s-au obţinut histogramele din figura 12.23. Precizaţi cărei instrucţiuni i se datorează această imagine . . . neterminată!

R.

```
380 PLOT j+1, i
```

în loc de

```
380 PLOT j, i
```

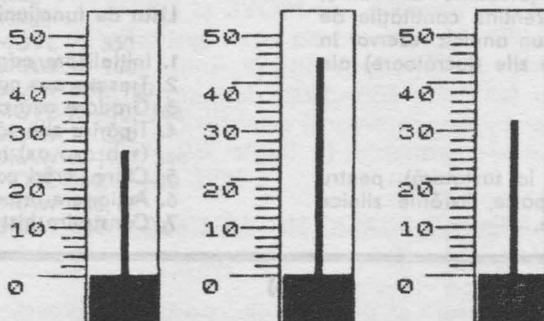


Fig. 12.23. Histograme cu . . . greşeli.

☐ Trasarea histogramelor în 2D prin metoda „linie cu linie" cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM

Vom relua problema din exemplul precedent complicînd-o după cum urmează.

Pentru trei rezervoare, se citesc de la tastatură cantitățile de combustibil livrate zilnic beneficiarilor. Se dorește ca, pentru fiecare rezervor în parte, să se traseze, printr-o metodă mai rapidă – linie cu linie, o histogramă reprezentînd cantitățile de combustibil livrate în fiecare din cele șase zile lucrătoare ale săptămînii.

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată, figura 12.24.

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Variabile de intrare	Variabile de stare	Variabile de ieșire
	t : livrări benzină, pe zile d : număr rezervor x, y : coordonate vîrf k, i, r, q1, q, p : variabile de lucru	t : livrări benzină

a)

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Descrierea programului

Programul afișează succesiv trei familii de histograme (alcătuite din linii), fiecare familie reprezentînd cantitățile de benzină livrate dintr-un anumit rezervor în fiecare din cele șase zile (lucrătoare) ale săptămînii.

Intrări

Se citesc (de la tastatură), pentru fiecare rezervor în parte, livrările zilnice de benzină efectuate.

Ieșiri

Trei histograme

Lista de funcțiuni ale programului

1. Inițializare origine sistem axe xoy
2. Trasare axe cu săgeți în vîrf
3. Gradare axă oy
4. Tipărire zile săptămînă (sub axa ox) în clar
5. Citire livrări combustibil, pe zile
6. Acționare orice tastă
7. Construire histogramă din linii

b)

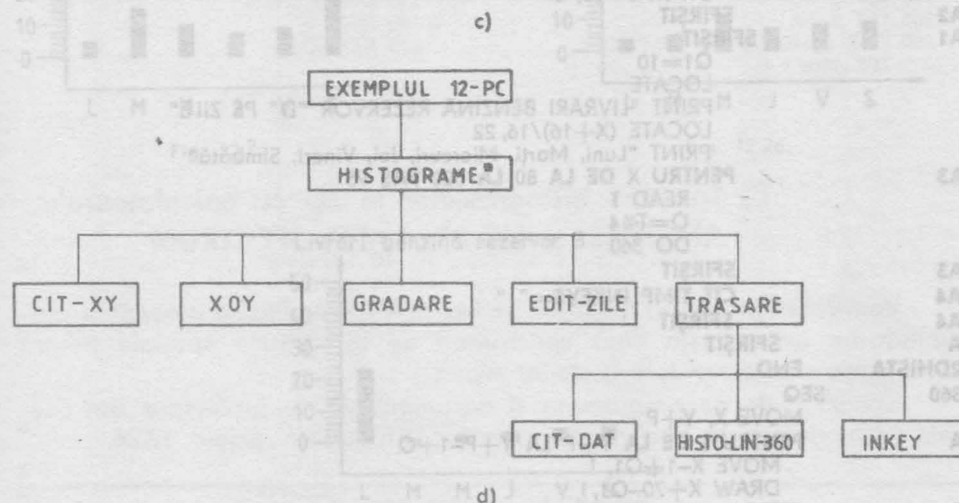
Fig. 12.24. Modulele de analiză și proiectare structurată: *a)* tabela de variabile; *b)* specificații de programare;

ALOCAREA FUNCŢIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Modul	Funcţiuni
CIT – XY	1
XOY	2
GRADARE	3
EDIT – ZILE	4
CIT – DAT	5
HISTO-LIN-360	7
INKEY	6

c)



d)

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (AMSTRAD)

```
2DHISTA  SEQ
         READ X, Y
A        PENTRU D DE LA 1 LA 3
           CLS
           K=0
           X=80
           MOVE X, 350
           DRAW X, 100
           DRAW 640, 100
           MOVE X -10, 340
           DRAW X, 350
           DRAW X +10, 340
           MOVE 630, Y+10
           DRAW 640, Y
           DRAW 630, Y-10
```

e)

Fig. 12.24. *c)* alocarea funcţiunilor de prelucrare; *d)* diagrama de structură; *e)* pseudocodul programului BASIC-AMSTRAD.

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```
A1          PENTRU I DE LA Y LA Y+200 PAS 8
              MOVE X, I
              R=(I-Y)/40
A2            DACĂ INT(R)=R
                DRAWR -20, 0
                LOCATE (X-70)/16, (400-I)/16
                PRINT K
                K=K+10
A2            IN CAZ CONTRAR
                DRAWR -10, 0
A2            SFÎRŞIT
A1          SFÎRŞIT
              Q1=10
              LOCATE
              PRINT "LIVRĂRI BENZINĂ REZERVOR "D" PE ZILE"
              LOCATE (X+16)/16, 22
              PRINT "Luni, Marţi, Miercuri, Joi, Vineri, Sîmbătă"
A3            PENTRU X DE LA 80 LA 620 PAS 95
                READ T
                Q=T*4
                DO 360
A3            SFÎRŞIT
A4            CÎT TIMP INKEY$=" "
A4            SFÎRŞIT
A           SFÎRŞIT
2DHISTA    END
360        SEQ
              MOVE X, Y+P
A           PENTRU I DE LA Y+P LA Y+P-1+Q
              MOVE X-1+Q1, I
              DRAW X+70-Q1, I
A           SFÎRŞIT
360        END
```

Fig. 12.24. *e)*

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 264

Lista programului este prezentată în volumul 2, pag. 264). De urmărit în cadrul programului: trasarea (cu **MOVE** şi **DRAW**) sistemului de coordonate xoy (v. liniile 70–116); scalarea axei oy (v. liniile 120–220); tipărirea (sub axa ox) numelor zilelor săptămînii (v. liniile 240–250); introducerea dinamică a livrărilor (v. liniile 260–330); acţionarea oricărei taste în vederea trasării histogramei pentru rezervorul următor (v. linia 335); construirea propriu-zisă din linii, a histogramelor (v. liniile 360–440).

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 267

Diferenţele de scriere a programului apar la utilizarea setului de instrucţiuni grafice **(PLOT, DRAW)** specifice calculatoarelor personale HC-85, TIM S şi SPECTRUM.

După ce consultaţi specificaţiile de programare şi documentaţia de proiectare a programului încercaţi să continuaţi, (singuri) cu codificarea programului. Dacă nu reuşiţi consultaţi lista programului din vol. 2, pag. 267.

În figurile 12.25, 12.26 şi 12.27 sînt ilustrate histogramele realizate ,,linie cu linie" pentru livrările de benzină din cele trei rezervoare.



Fig. 12.25.



Fig. 12.26.

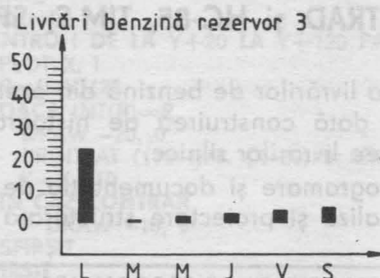


Fig. 12.27.

Aplicaţie. Introduceţi şi executaţi următoarele programe:

a)

```
5 BORDER 7 : PAPER 7 : CLS : INK 0
10 LET x=INT (RND*256)
20 LET y=INT (RND*176)
30 LET l=INT (RND*256)
40 LET m=INT (RND*176)
50 LET u=15 : LET v=7
60 DEF FN r(x)=INT (RND * x)
100 GO SUB 1000
115 LET num=num-1
116 IF num=0 THEN GO SUB 1000
120 PLOT x, y
130 DRAW l-x, m-y
140 IF x+a>255 OR x+a<0 THEN
    LET a=-a
150 IF y+b>175 OR y+b<0 THEN
    LET b=-b
160 IF l+c>255 OR l+c<0 THEN
    LET c=-c
170 IF m+d>175 OR m+d<0 THEN
    LET d=-d
180 LET x=x+a : LET y=y+b
190 LET l=l+c : LET m=m+d
200 LET cont=FN r(200)
220 IF cont=l THEN RUN
230 GO TO 115
1000 LET a=FNr(u)-v
1010 LET b=FNr(u)-v
1020 LET c=FNr(u)-v
1030 LET d=FNr(u)-v
1040 LET num=FNr(20)+10
1050 RETURN
```

b)

```
10 READ y
20 DATA 20
30 CLS
35 LET p=0; LET q1=0
40 FOR x=40 TO 200 STEP 80
50    PLOT x, y
60    DRAW 30,0
70    DRAW 0,130
80    DRAW -30,0
90    DRAW 0,-130
100   LET q=20
```

```
110 LET k=0
115 LET q1=28
120 GO SUB 340
130 FOR i=y+20 TO y+120 STEP 4
140 PLOT x, i
150 LET r=(i-y-20)/20
160 IF INT (r)<>r THEN GO TO 22
170 DRAW -20,0
180 PRINT AT (175-i)/8, (x-30)/8
200 LET k=k+10
210 GO TO 230
220 DRAW -10,0
230 NEXT i
240 NEXT x
250 LET q1=8
255 LET d=0
260 FOR x=40 TO 200 STEP 80
262 LET d=d+1
265 PRINT AT 1,1; "cantitatea in tone
    rezervor"; d;
280 INPUT t
290 LET p=20
300 LET q=t*2
310 GO SUB 340
315 PRINT AT 1,1;"
                                   "
320 NEXT x
325 GO TO 325
330 STOP
340 PLOT x, y+p
350 FOR i=y+p TO y+p-1+q
360 PLOT x+(30-q1)/2, i
370 DRAW q1, 0
400 NEXT i
420 RETURN
```

□ Trasarea histogramelor în 3D cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM

Vom relua problema livrărilor de benzină din exemplul precedent propunîndu-ne de această dată construirea de histograme spațiale pentru vizualizarea și interpretarea livrărilor zilnice.

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată figura 12.28.

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLUL 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Descrierea programului	Lista de funcțiuni ale programului
Programul afișează succesiv trei familii de histograme spațiale (alcătuite din linii), fiecare familie reprezentînd cantitățile de benzină livrate dintr-un anumit rezervor în fiecare din cele șase zile (lucrătoare) ale săptămînii. **Intrări** Se citesc (de la tastatură), pentru fiecare rezervor în parte, livrările zilnice de benzină efectuate. **Ieșiri** Trei histograme spațiale.	1. Inițializare origine sistem axe xoy 2. Trasare axe cu săgeți în vîrf 3. Gradare axă Oy 4. Tipărire nume zile săptămînă (sub axa Ox) în clar 5. Citire livrări combustibil, pe zile 6. Acționare orice tastă 7. Trasare linie orizontală de lățime 50 puncte 8. Trasare linie înclinată la 45° față de linia orizontală 9. Translatare linie frîntă pe o direcție paralelă cu Oy 10. Completare ultima linie frîntă trasată pînă la un paralelogram.

a)

Fig. 12.28. Modulele de analiză și proiectare structurată: a) specificații de programare;

PSEUDOCODUL
Nume program: EXEMPLUL 12-PC (HC-85, TIMS, SPECTRUM)

```
HC30   SEQ
         READ X, Y
A        PENTRU D DE LA 1 LA 3
           X=40
           CLS
           K=0
           P=0
           Q1=0
           PLOT X, Y
           DRAW 200, 0
           DRAW -5, 5
           PLOT X+200, Y
           DRAW -5, -5
           PLOT X, Y
           DRAW 0, 137
           DRAW -5, -5
           PLOT X, 137+Y
           DRAW 5, -5
           Q=20
           K=0
           Q1=28
A1       PENTRU I DE LA Y+20 LA Y+120 PAS 4
           PLOT X, I
           R=(I-Y)/20
B          DACĂ INT(R)=R
             DRAW -20, 0
             PRINT AT (175-I)/8, (X-30)/8; K
             K=K+10
B          ÎN CAZ CONTRAR
                 DRAW -10, 0
B          SFÎRŞIT
A1       SFÎRŞIT
           Q1=8
           PRINT AT 1, 1; "LIVRĂRI BENZINĂ REZERVOR";
           PRINT AT (175-Y)/8+1, X/8; "
           L M M J V S "
C          PENTRU X DE LA 40 LA 190 PAS 30
             READ T
             P=20
             Q=T*2
             DO 340
C          SFÎRŞIT
D          CÎT TIMP INKEY$=" "
D          SFÎRŞIT
HC3D   END
340    SEQ
A        PENTRU I DE LA Y+P LA Y+P-1+Q
           PLOT X+(30-Q1)/2, I
           DRAW Q1, 0
           DRAW 5, 5
A        SFÎRŞIT
         DRAW -Q1, 0
         DRAW -5, -5
340    END
```

b)

Fig. 12.28. *b)* pseudocodul programului BASIC HC-85, TIM S, SPECTRUM.

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```
3D    SEQ
         READ X, Y
A        PENTRU D DE LA 1 LA 3
            CLS
            K=0
            X=80
            MOVE X, 350
            DRAW X, 100
            DRAW 640, 100
            MOVE X-10, 340
            DRAW X, 350
            DRAW X+10, 340
            MOVE 630, Y+10
            DRAW 640, Y
            DRAW 630, Y-10
A1          PENTRU I DE LA Y LA Y+200 PAS 8
               MOVE X, I
               R=(I-Y)/40
A2             DACĂ INT(R)=R
                  DRAWR -20, 0
                  LOCATE (X-70)/16, (400-I)/16
                  PRINT K
                  K=K+10
A2             ÎN CAZ CONTRAR
                  DRAWR -10, 0
A2             SFÎRŞIT
A1          SFÎRŞIT
            Q1=10
            LOCATE, 1, 1
               PRINT "LIVRĂRI BENZINĂ REZERVOR "D" PE ZILE"
            LOCATE (X+16)/16, 22
            PRINT "Luni Marţi Mierc. Joi Vineri Sîmb."
B        PENTRU X DE LA 80 LA 620 PAS 95
            READ T
            Q=T * 4
            DO 360
            DRAW X+Q1+9, I+9
            DRAW X-1+Q1, I-1
B        SFÎRŞIT
C        CÎT TIMP INKEY$=""
C        SFÎRŞIT
A        SFÎRŞIT
3D       END
360      SEQ
            MOVE X, Y+P
A        PENTRU I DE LA Y+P LA Y+P-1+Q
            MOVE X-1+Q1, I
            DRAW X+70-Q1, I
            DRAW X+70-Q1+10, I+10
A        SFÎRŞIT
360      END
```

c)

Fig. 12.28. *c)*

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 265

Lista programului se găseşte în volumul 2, pag. 265.

De urmărit în cadrul programului:

10– 20 iniţializare origine sistem axe
70–116 trasare axe (cu săgeţi în vîrf)
120–220 construirea unei scări de măsură
260–330 generare histograme în 2D pentru cele trei rezervoare
360–440 generare histogramă spaţială (3D)

Scalarea

Deşi problema v-a mai fost prezentată (niciodată însă cu explicaţiile noastre) considerăm că este util să aducem cîteva precizări.

Cunoaşteţi rezoluţia AMSTRAD-ului! 640×400 puncte cu punctul de coordonate (0, 0) în colţul din stînga jos al ecranului. Totodată, ecranul mai poate fi privit şi ca un sistem x′o′y′ (punctul 0′ de coordonate (0, 0) se află în colţul din stînga sus) mărimea ecranului fiind de 40×25 caractere. Rezultă că, un caracter este alcătuit din (640/40)×(400/25)=16×16 pixeli (puncte). Pe baza acestor considerente putem stabili următoarele formule de trecere de la un sistem de coordonate la altul.

$$y' \cdot 16 = 400 - y \tag{11}$$

$$x' \cdot 16 = x \tag{12}$$

de unde, rezultă

$$y' = \frac{400-y}{16} \tag{13}$$

$$x' = \frac{x}{16} \tag{14}$$

Relaţiile (13) şi (14) justifică utilizarea instrucţiunilor din liniile 180 şi 190.

Cît priveşte reprezentarea livrărilor de combustibil am făcut următoarele ipoteze: a) cantitatea maximă de combustibil ce poate fi reprezentată este de 50 tone; b) pentru o tonă de combustibil folosim un spaţiu de patru puncte ecran (v. linia 310). Deci, pentru a reprezenta 10 tone avem nevoie de un spaţiu de 40 puncte (v. ciclul de instrucţiuni din liniile 120–220).

Cu instrucţiunea

```
150 IF INT(r)<>r THEN 210
```

selectăm punctele a căror diviziune este multiplu de 10 (exemplu: 0, 10, 20, 30, 40, 50) semnalate printr-o liniuţă mai lungă (v. linia 160) faţă de cele care semnalează cantităţile de combustibil divizibile cu 2 (v. linia 210).

Observaţie. Deoarece scalarea s-a realizat din două în două tone ciclul **FOR–NEXT** din liniile 120–220 are pasul 8 (o tonă se reprezintă prin 4 puncte).

Histograma spaţială

Cît priveşte subrutina 360 aducem următoarele precizări suplimentare. Ea reprezintă varianta unei subrutine cunoscute de desenare a histogramelor. De notat că în acest program paramentul p nu este iniţializat în mod explicit, deci implicit va lua valoarea zero.

Instrucţiunile din liniile 380–390 trasează o linie de lăţime $x+70-q1-x+1-q1=70-2q1$ puncte situate la o înălţime de i puncte de baza ecranului respectînd condiţia:

$$y+p \leqslant i \leqslant y+p-1+q \quad (15)$$

Deoarece q1=10 (v. linia 230) în cazul de faţă lăţimea histogramei va fi de 50 puncte (70–2×10 puncte).

Instrucţiunea din linia 395 desenează în continuarea acestei linii o altă linie înclinată faţă de aceasta cu 45°.

În figura 12.29 am reprezentat linia frîntă rezultată, indicînd coordonatele absolute ale punctelor ce o determină.

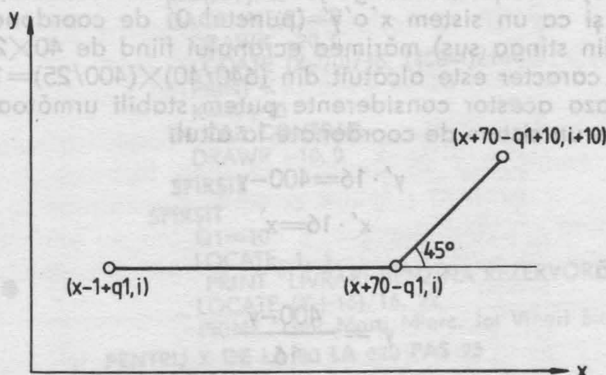


Fig. 12.29.

Ciclul **FOR-NEXT** cuprins între liniile 370 şi 420 nu face decît să translateze această linie frîntă pe o direcţie paralelă cu axa Oy rezultînd pe ecran o figură asemănătoare cu cea din fig. 12.30 (a).

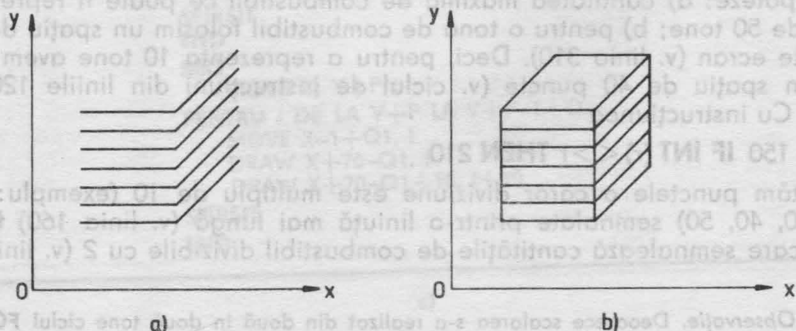


Fig. 12.30.

Instrucţiunile din liniile 325 şi 326 pun „capac" acestei figuri; de fapt se completează pînă la un paralelogram ultima linie frîntă (v. fig. 12.30 (b) trasată cu ciclul **FOR-NEXT** din liniile 370–420.

Aplicaţie. Introduceţi şi executaţi următoarele programe BASIC-AMSTRAD:

a)
```
100 REM
110 MODE 2
120 DIM upper (640), lower (640)
130 INPUT "Continuaţi/Stop (Y/N)",
    ans$
140 IF LEFT$ (ans$, 1)="y" THEN
    lne=–1 ELSE lne=0
150 centx=320 : centy=200
160 viewx=275 : viewz=120
170 REM
180 ht=40 : ohm=0.043
190 :
200 FOR i=1 TO 640
210   upper (i)=0
220   lower (i)=1000
230 NEXT i
240 :
250 FOR z=viewz–1 TO–viewz+1
    STEP –5
260 lowx=INT(viewx*SQR (l–z*z/
    viewz/viewz)+0.5)
270 x=–lowx
280 y=ht*SIN (ohm*SQR(x*x+z*z))
290 x1=x+centx+z
300 y1=INT (400–(centy+y+z/2)+0.5)
310 FOR x=–lowx+1 TO lowx–1
320 y=ht*SIN(ohm*SQR(x*x+z*z))
330 x2=centx+x+z
340 y2=INT (400–(centy +y+z/2)+0.5)
350 IF lne=0 THEN GOTO 390
360 IF y2<lower (x2) THEN GOSUB
    460
370 IF y2>upper (x2)
    THEN upper (x2)=y2 : MOVE
    x1, y1 : DRAW x2, y2
380 GOTO 400
390 MOVE x1, y1 : DRAW x2, y2
400 x1=x2
410 y1=y2
420 NEXT x
430 NEXT z
440 END
450 :
460 lower (x2)=y2
470 IF upper (x2)=0 THEN upper
    (x2)=y2
480 RETURN
```

b)
```
100 REM Histograme 3D
110 MODE 0
120 k1=5 : k2=20 : k3=10
130 REM
140 p=100 : q=100
150 MOVE p, q : DRAWR 300,
    300*k1/k3,1 : DRAWR 200,
    –200*k1/k2,1
160 DRAWR–300, –300*k1/k3,1 :
    DRAWR –200, 200*k1/k2,1
170 FOR i=1 TO 3
180   MOVE p, i*50+q : DRAWR
    300, 300*k1/k3,1 : DRAWR 200,
    –200*k1/k2,1
190 NEXT i
200 MOVE p, q : DRAWR 0, 150,1
210 MOVER 300, 300*k1/k3 : DRAWR
    0, –150,1
220 MOVER 200, –200*k1/k2 : DRAWR
    0,150,1
230 :
240 REM
250 ik=2 : off=80
260 READ h : x=p+40 : y=q–8 :
    GOSUB 450
270 FOR j=1 TO 3
280 READ h : x=x+off : y=x+off*
    k1/k3 : GOSUB 450
290 NEXT i
300 REM
310 ik=3 : off=80
320 READ h : x=p+190 : y=q–20 :
    GOSUB 450
330 FOR j=1 TO 3
340   READ h : x=x+off : y=y+off*
    GOSUB 450
350 NEXT j
360 REM
370 ik=4 : off=80
380 READ h : x=p+240 : y=q–32 :
    GOSUB 450
390 FOR j=1 TO 3
400   READ h : x=x+off : y=y+off*
    k1/k3 : GOSUB 450
410 NEXT j
420 :
430 z$=INKEY$ : IFz$="" THEN 430
440 END
450 REM MOVE x, y : DRAWR 0, h, ik
460 FOR p=0 TO k2 STEP 2
470   MOVE p+x, y–p*k1/k2
480   DRAWR 0, h, ik
490   DRAWR k3, k1, ik
500 NEXT p
510 :
520 FOR p=k3 TO 0 STEP –2
530   MOVE x+k2+p, y–(k3–p)*k1/k3
540   DRAWR 0, h, ik
550 NEXT p
560 :
570 MOVE x+k2, y–k3*k1/k3
580 DRAWR 0,h+2,0
590 DRAWR k3, k1, 0
600 MOVE x, y+h : DRAWR k2, –k1,0
610 RETURN
```

```
620 DATA 100, 80, 120, 100, 150, 60,
    100
630 DATA 100, 140, 75, 126, 150
```

c)
```
100 REM Histograme
110 MODE 1
120 SYMBOL 244, 126, 126, 126, 126,
    126, 126, 126, 126
130 SYMBOL 245, 24, 24, 24, 24, 24,
    24, 24, 24
140 SYMBOL 246, 0, 0, 0, 0, 0, 0, 0,
    255
150 DEF FNprt$(a)=RIGHT$ ("
    "+STR$(a), 3)
160 INPUT "Nr. bare "(<30)", bars
170 DIM value (bars)
180 maximum=0
190 FOR i=1 TO bars
200   LOCATE 1,5
210   INPUT "Valori", value (i)
220   IF value (i) > maximum THEN
      maximum= value (i)
230 NEXT i
240 :
250 CLS
260 scale=1
270 IF maximum>20 THEN scale=
    =maximum/20
280 :
290 FOR i=1 TO bars
300   value (i)=INT (value(i)/scale)
310 NEXT i
320 :
330 FOR i=1 TO 20 STEP 2
340   LOCATE 1, 22-i : PRINT FNprt$
      (INT(i*scale))
350 NEXT i
360 :
370 FOR i=1 TO 23
380   LOCATE 4, i : PRINT CHR$ (245);
390 NEXT i
400 :
410 FOR i=1 TO bars+4
420   LOCATE 4+i, 22 : PRINT CHR$
      (246);
430 NEXT i
440 :
450 FOR i=1 TO bars
460   FOR j=1 TO value(i)
470     LOCATE 7+i,22-j : PRINT
        CHR$ (244);
480   NEXT j
490 NEXT i
500 z$=INKEY$ : IF z$=" "
    THEN 500
```

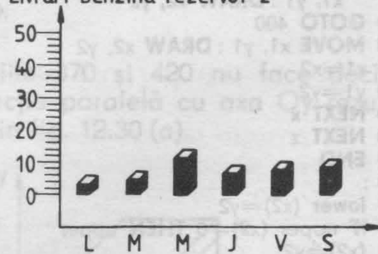
Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

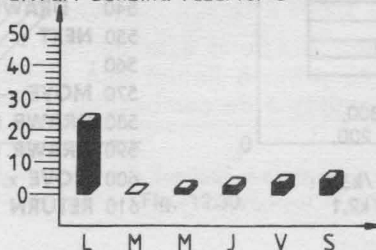
vol. 2, pag. 267

Diferenţele privind realizarea histogramelor spaţiale pe calculatoarele HC-85, TIM S şi SPECTRUM ţin atît de rezoluţia ecranului cît şi de setul de instrucţiuni grafice disponibile pe aceste tipuri de calculatoare.

În urma execuţiei programului s-au generat următoarele histograme·







Aplicație. Introduceți și executați următoarele programe:

a)
```
10 FOR y=10 TO-10 STEP-1
20 IF y<>10 AND y<>-10 AND y>-1 THEN PRINT "";
30 PRINT y; TAB 4;
40 FOR x=-10 TO 10
50 LET k=y-x*x/2+7
60 PRINT ("0" AND k>=.5)+("." AND k<.5);
70 NEXT x
80 PRINT
90 NEXT y
100 PRINT TAB 4; ".9.7.5.3.1.1.3.5.7.9."
```

b) Inlocuiți liniile 60 și 70 cu instrucțiunile

```
60 LET k=SQR (ABS(y*x*2))-x
70 PRINT ("■" AND k<=.5)+("." AND k>.5);
```

c) Inlocuiți liniile 60 și 70 cu instrucțiunile

```
60 LET k=ABS (y+x)-x*x
70 PRINT ("■" AND k>=.5)+("." AND k<.5);
```

□ Trasarea histogramelor în 2D și 3D cu microcalculatoarele M118 și TPD

vol. 2, pag. 268

Particularități BASIC-80

Microcalculatoarele M118 și TPD de construcție românească, spre deosebire de microcalculatorul JUNIOR sînt prevăzute și cu opțiune grafică (rezoluția ecranului este de 512×256 puncte). Setul de instrucțiuni grafice disponibile pe cele două tipuri de microcalculatoare este următorul: **VIEWPORT** (coordonatele au valori cuprinse între 0–511 și 0–255), **WINDOW, MOVE, RMOVE, DRAW, RDRAW, ROTATE** (rotește o linie trasată pe ecran în coordonate relative).

Pregătirea ecranului (ștergerea altor caractere și comutarea din mod defilare în mod pagină) se realizează cu instrucțiunile BASIC-80:

```
SIR$=CHR$(27)+"|"+CHR$(27)+"2"
PRINT SIR$
```

pentru varianta de monitor MON V3.6, și cu instrucțiunile:

```
SIR$=CHR$(27)+"0"
PRINT SIR$
```

pentru varianta de monitor V4.2.

Observație. Codurile "|" și "2" realizează respectiv ștergerea ecranului și comutarea din mod defilare (ecranul defilează) în mod pagină (ecran fix).

Grafica realizată pe cele două microcalculatoare permite utilizarea modurilor de lucru **ALFA** și **GRAFIC.** Reprezentarea informațiilor în modul de lucru **ALFA** se realizează sub formă alfanumerică (v. caracterele alfanu-

merice) pe 24 de linii ecran a cîte 80 caractere fiecare. De notat că poziționarea cursului se face în raport cu colțul din stînga sus.

Reprezentarea informațiilor în modul de lucru ALFA se realizează cu instrucțiunea **PRINT** al cărei format general este:

Format general
PRINT CHR$**(27)+"1"+**CHR$(X+32)+**CHR$**(Y+32); "⟨text alfa⟩"

în care ⟨X⟩ și ⟨Y⟩ reprezintă coordonatele cursorului la un moment dat. Valorile celor două coordonate se supun următoarelor restricții: $x \leqslant 80$; $y \leqslant 24$.

Reprezentarea în modul de lucru GRAFIC se realizează la nivel de bit.

Remarci privind utilizarea versiunii 5.2 a interpretorului BASIC-80.

- Pentru a schimba modul de generare a vectorilor (numai pentru versiunea 5.2 a interpretorului BASIC-80) puteți utiliza instrucțiunile:

  POKE &H520A, &HC1 (funcționare normală)
  POKE &H520A, &HC8 (ștergerea vectorului)

- In mod normal funcția **GCLEAR** are ca efect ștergerea ferestrei fizice selectate. Se poate însă umple fereastra logică! Utilizați următoarea secvență de instrucțiuni

  POKE &H5268, &H81
  POKE &H52A7, &H41
  POKE &H52B4, &H81

- Pentru readucerea la normal a instrucțiunii **GCLEAR** utilizați secvența de instrucțiuni

  POKE &H5268, &H38
  POKE &H52A7, &H48
  POKE &H52B4, &H88

Inapoi la program

Vom relua și în cadrul acestui program (operațional pe microcalculatoarele M118 și TPD) problema livrărilor de benzină afișînd de această dată sub forma unor histograme „Situația livrărilor zilnice la trei rezervoare". De notat că programul generează într-un mod care cu siguranță o să vă încînte ochiul, atît histograme în spațiul 2D cît și histograme în spațiul cu trei dimensiuni (3D).

De această dată vă punem la dispoziție numai două „piese" de analiză-proiectare și anume: ieșirile programului (histogramele în 2-D și 3-D (v. fig. 12.31, fig. 12.32) și pseudocodul (v. tabelul 12.3) după care am realizat codificarea.

Ne permitem să vă atragem atenția asupra acestui program (realizat cu versiunea 5.2), puțin mai dificil decît precedentele.

Lista programului se află în vol. 2, pag. 268. Pentru înțelegerea lui revedeți și particularitățile instrucțiunilor grafice expuse mai sus.

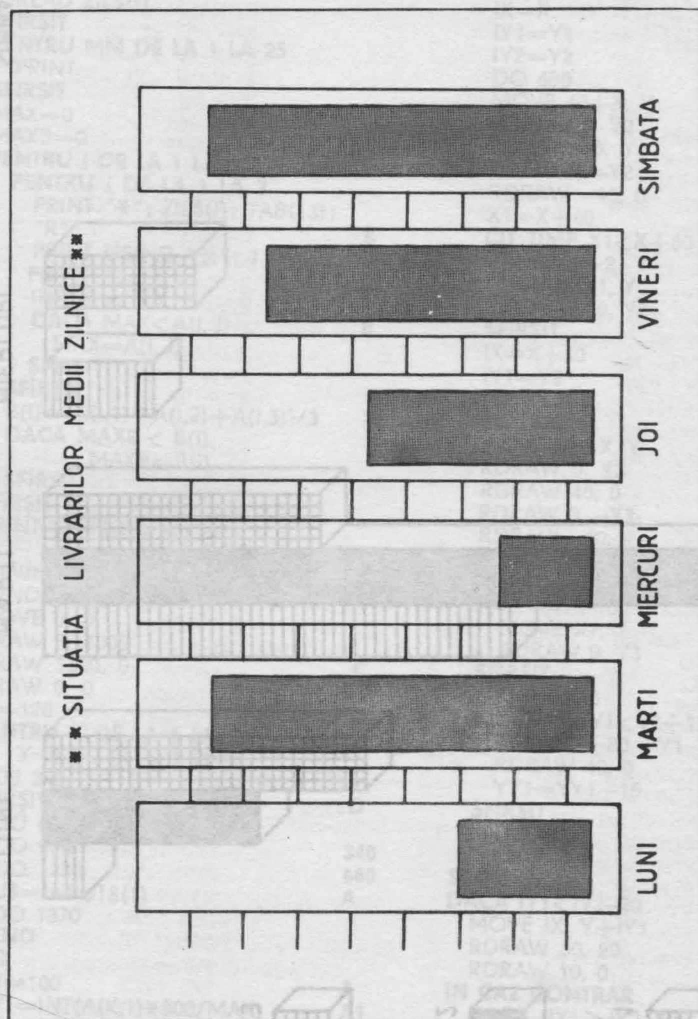

** SITUATIA LIVRARILOR MEDII ZILNICE **
LUNI
MARTI
MIERCURI
JOI
VINERI
SIMBATA

Fig. 12.31.

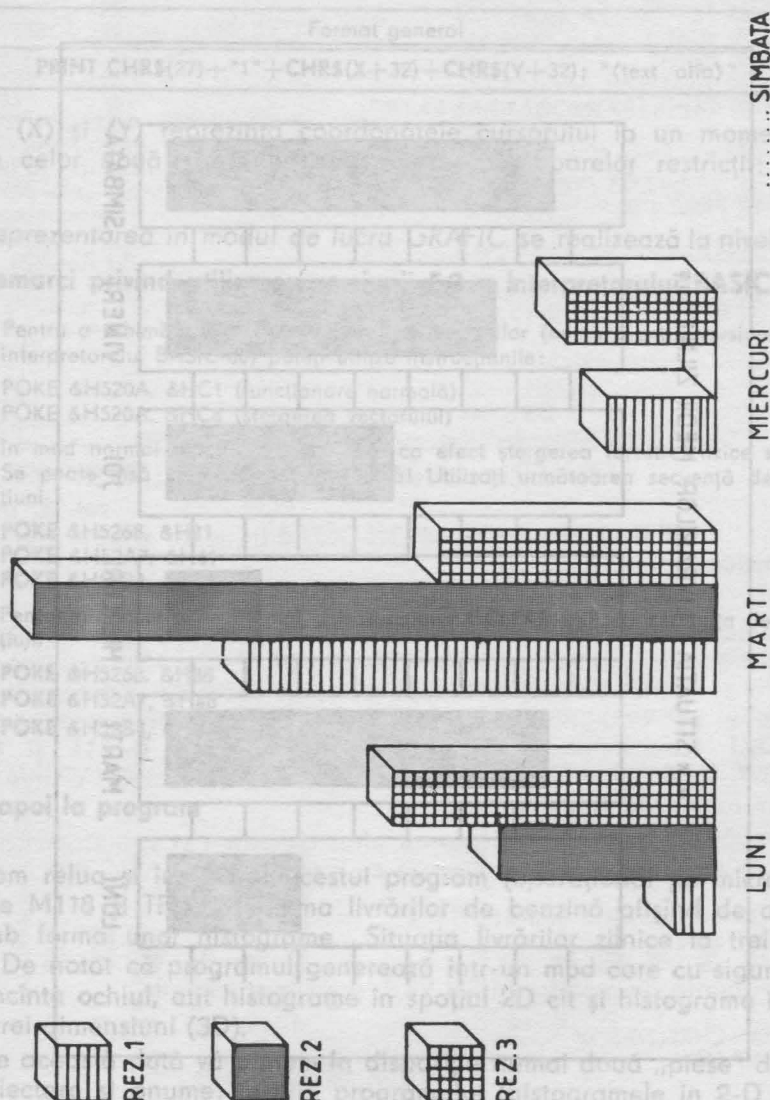

■■■SITUATIA LIVRARILOR ZILNICE LA TREI REZERVOARE■■■
LUNI
MARTI
MIERCURI
..........SIMBATA
REZ1
REZ2
REZ3

Fig. 12.32.

Tabelul 12.3

PSEUDOCODUL

Nume program: EXEMPLUL 12 – m

```
EX13M118 SEQ
A        PENTRU I DE LA 1 LA 6
            READ ZIL$(I)
A        SFÎRŞIT
B        PENTRU MM DE LA 1 LA 25
            PRINT
B        SFÎRŞIT
         MAX=0
         MAXB=0
A1       PENTRU I DE LA 1 LA 6
B1         PENTRU J DE LA 1 LA 3
             PRINT "*"; ZIL$(I); TAB(13);
             "R"
             PRINT USING "#"; J
             PRINT "..."
             INPUT A(I, J)
B2           DACĂ MAX< A(I, J)
               MAX=A(I, J)
B2           SFÎRŞIT
B1         SFÎRŞIT
           B(I)=(A(I,1)+A(I,2)+A(I,3))/3
B3         DACĂ MAXB < B(I)
                MAXB=B(I)
B3         SFÎRŞIT
A1       SFÎRŞIT
         PRINT CHR$(27)+"I"+
               +CHR$(27)+"2"
         VIEWPORT 0, 132, 0, 99
         WINDOW 0, 1320, 0, 1000
         MOVE 0, 0
         DRAW 0, 1000
         DRAW 1320, 0
         DRAW 0, 0
         X0=120
D        PENTRU K DE LA 1 LA 6
            X=XO+(K-1) * 200
            DO 340
D        SFÎRŞIT
         DO 840
         DO 950
         DO 1230
         U$=INPUT$(1)
         DO 1370
EX13M118 END
340      SEQ
         Y=100
         Y1=INT(A(K,1)*800/MAX)
         Y2=INT(A(K,2)*800/MAX)
         Y3=INT(A(K,3)*800/MAX)
         MOVE X, Y
         RDRAW 0, Y1
         RDRAW 40, 0
         RDRAW 0, -Y1
         RDRAW -40, 0
         YY1=Y
A        CÎT TIMP YY1< Y1+99
            YY1=YY1+10
            MOVE X, YY1
            RDRAW 40, 0
A        SFÎRŞIT
         IX=X
         IY1=Y1
         IY2=Y2
         DO 680
         MOVE 40+X, Y
         RDRAW 0, Y2
         RDRAW 40, 0
         RDAW 0, -Y2
         RDRAW -40, 0
         X1=X+40
B        CÎT TIMP X1<X+80
           X1=X1+2
           MOVE X1, Y
           RDRAW 0, Y2
B        SFÎRŞIT
         IX=X+40
         IY1=Y2
         IY2=Y3
         DO 680
         MOVE 80+X, Y
         RDRAW 0, Y3
         RDRAW 40, 0
         RDRAW 0, -Y3
         RDRAW -40, 0
         X1=X+80
C        PENTRU MM DE LA 1 LA 3
           X1=X1+10
           MOVE X1, Y
           RDRAW 0, Y3
C        SFÎRŞIT
         YY1=Y+15
D        CÎT TIMP YY1<Y3+100
           MOVE X+80, YY1
           RDRAW 40, 0
           YY1=YY1+15
D        SFÎRŞIT
         DO 800
340      END
680      SEQ
A        DACĂ IY1<IY2-20
           MOVE IX, Y+IY1
           RDRAW 30, 20
           RDRAW 10, 0
A        ÎN CAZ CONTRAR
A1         DACĂ (IY1 ≥IY2-20)
              OR (IY1 ≤IY2)
              YY1=INT (37-(IY2-IY1))
              MOVE IX, Y+IY1
              RDRAW 30, 20
              RDRAW YY1, 0
A1         ÎN CAZ CONTRAR
              YY1=IY1-IY2
              MOVE IX, Y+IY1
              RDRAW 30, 20
              RDRAW 40, 0
              RDRAW -30, 20
```

Tabelul 12.3 (continuare)

PSEUDOCODUL

Nume program: EXEMPLUL 12 – m

```
          RMOVE 30, 20
          RDRAW 0, -YY1
A1      SFÎRŞIT
A     SFÎRŞIT
680   END
840   SEQ
      PRINT CHR$(27)+"1"+CHR$(45)+
      +CHR$(33)
      YY1=41
      DO 930
      PRINT "Luni"
      YY1=53
        DO 930
        PRINT "Marţi"
        YY1=63
        DO 930
        PRINT "Miercuri"
        YY1=78
        DO 930
        PRINT "Joi"
        YY1=88
        DO 930
        PRINT "Vineri"
        YY1=100
        DO 930
        PRINT "Simbătă"
840   END
930   SEQ
        PRINT CHR$(27)+"1"+
        +CHR$(YY1)+CHR$(54)
930   END
950   SEQ
        MOVE 40, 850
        RDRAW 0, 40
        RDRAW 40, 0
        RDRAW 0, -40
        RDRAW -40, 0
        MOVE 40, 890
        RDRAW 30, 20
        RDRAW 40, 0
        RDRAW -30, 20
        YY1=10
        XX1=850+YY1
        RMOVE 30, 20
        RDRAW 0, -40
        RDRAW -30, -20
A     CÎT TIMP XX1<889
        MOVE 40, XX1
        RDRAW 40, 0
        XX1=XX1+YY1
A     SFÎRŞIT
      MOVE 40, 550
      RDRAW 0, 40
      RDRAW 40, 0
      RDRAW 0, -40
      RDRAW -40, 0
      MOVE 40, 590
      RDRAW 30, 20
      RDRAW 40, 0
      RDRAW -30, -20
      RMOVE 30, 20
      RDRAW 0, -40
      RDRAW -30, -20
      YY1=2
      XX1=42
B     CÎT TIMP XX1<79
        MOVE XX1, 550
        RDRAW 0, 40
        XX1=XX1+2
B     SFÎRŞIT
      MOVE 40, 250
      RDRAW 0, 40
      RDRAW 40, 0
      RDRAW 0, -40
      RDRAW -40, 0
      MOVE 40, 290
      RDRAW 30, 20
      RDRAW 40, 0
      RDRAW -30, -20
      RMOVE 30, 20
      RDRAW 0, -40
      RDRAW -30, -20
C     PENTRU MM DE LA 1 LA 3
        MOVE 40+MM * 10, 250
        RDRAW 0, 40
C     SFÎRŞIT
      YY1=265
D     CÎT TIMP YY1<285
        MOVE 40, YY1
        RDRAW 40, 0
        YY1=YY1+15
D     SFÎRŞIT
950   END
1230  SEQ
        YY1=37
        DO 1270
        PRINT "Rez. 1"
        YY1=44
        DO 1270
        PRINT "Rez. 2"
        YY1=51
        DO 1270
        PRINT "Rez. 3"
1230  END
1270  SEQ
        PRINT CHR$(27)+"1"+
        +CHR$(33)+CHR$(YY1)
1270  END
1290  SEQ
        POKE &H5268, &H81
        POKE &H52A7, &H41
        POKE &H52B4, &H81
1290  END
```

Tabelul 12.3 (continuare)

PSEUDOCODUL
Nume program: EXEMPLUL 12 – m

```
1330   SEQ
         POKE H5268, H88
         POKE H52A7, H48
         POKE H52B4, H88
1330   END
1370   SEQ
         PRINT CHR$(27)+"I"+
            +CHR$(27)+"2"
         VIEWPORT 0, 133, 0, 99
         WINDOW 0, 1320, 0, 1000
         DO 1290
         GCLEAR
         DO 1330
         Y=10
         X=13
A        PENTRU MM DE LA 1 LA 6
           VIEWPORT X, X+16, Y, Y+80
           GCLEAR
           X=13+MM * 20
A        SFÎRŞIT
         YY1=15
         X1=10
         X2=125
B        PENTRU MM DE LA 1 LA 8
           VIEWPORT X1, X2, YY1, YY1+1
           GCLEAR
           YY1=YY1+10
B        SFÎRŞIT
         DO 1290
C        PENTRU MM DE LA 1 LA 6
           X1=15+(MM-1) * 20
           X2=X1+12
           Y1=15
           Y2=INT(70 * B(MM)/MAXB)+15
           VIEWPORT X1, X2, Y1, Y2
           GCLEAR
C        SFÎRŞIT
           PRINT CHR$(27)+"1"+
              +CHR$(52)+CHR$(33)
           PRINT "Situaţia livrărilor medii
                                    zilnice"
           PRINT CHR$(14)
           YY1=43
           DO 930
           PRINT "Luni"
           YY1=55
           DO 930
           PRINT "Marţi"
           YY1=65
           DO 930
           PRINT "Miercuri"
           YY1=80
           DO 930
           PRINT "Joi"
           YY1=90
           DO 930
           PRINT "Vineri"
           YY1=101
           DO 930
           PRINT "Sîmbătă"
           PRINT CHR$(15)
           PRINT CHR$(27)+"1"+
              +CHR$(33)+CHR$(35)
           PRINT USING "##.#"; MAXB
           PRINT CHR$(27)+"1"+CHR$(34)+
              +CHR$(52); "0"
           U$=INPUT$(1)
           PRINT CHR$(27)+"2"
1370   END
```

Aplicaţie: Introduceţi şi executaţi următorul program:

```
10 W=.0314159265
20 D=3.14159265/4
30 DIM A$ (1, 61), Z$(100), X(100), Y(100)
40 FOR I=1 TO 100 : T=I*8.6
50    X(I)=(100-T/10)*COS (W*T+D)+125
60    Y(I)=20*SIN(62*W*T)+20+T/5
70 NEXT I
90 J=29 : GOSUB 300
100 FOR I=1 TO 61 : A$ (1,I)=" " : NEXT I
105 J=J-1
110 FOR I=1 TO 100
120    IF Y (I)>(J+.5)*256/36 THEN 200
130 NEXT I
135 IF J<6 THEN A$ (1,30)="H"
140 FOR I=1 TO 61 : PRINT A$ (1,I); : NEXT I : PRINT
150 IF J<1 THEN 500
160 GO TO 100
200 K=INT(X(I)/256*60+0.5) : A$(1,K)="*"; Y(I)=Y(I)-300
220 GOTO 130
```

```
300 PRINT TAB (28) "*"
310 PRINT TAB (27) "*  *"
320 PRINT TAB (25) "* *"
330 PRINT TAB (27) "* *"
340 PRINT TAB (28) "*"
350 RETURN
500 END
```

Observaţie. Programul desenează un brad înalt.

TEMA 12

☐ Răspundeţi prin DA sau NU la următoarele întrebări:

– Instrucţiunile **MOVE** şi **DRAW** se deosebesc de instrucţiunile **RMOVE** şi **RDRAW** prin faptul că X şi Y nu sînt raportate la origine.

– O imagine monocromă (alb-negru) poate fi produsă cu ajutorul unei reţele rectangulare de puncte ale unui ecran prin iluminarea sau neiluminarea unor puncte.

– Instrucţiunile pentru poziţionarea spotului sînt aceleaşi pentru toate versiunile (dialectele) BASIC.

– Pentru a programa o imagine grafică pe ecran cu caractere mozaic nu este necesar să se precizeze codul caracterului grafic ce urmează a fi utilizat.

– Reprezentarea unui caracter grafic pe ecran se face cu ajutorul unei instrucţiuni

POKE X, Y

– Instrucţiunea **WINDOW** defineşte spaţiul utilizator.

– Cu instrucţiunea **CIRCLE** se pot genera cercuri în limbajul BASIC-aMIC.

– Setul de instrucţiuni grafice corespunzătoare limbajului BASIC-SPECTRUM nu cuprinde instrucţiunea **MOVE**.

– Histogramele constituie un preţios ajutor în special pentru interpretarea rezultatelor.

☐ Înlocuiţi cuvintele care lipsesc din următoarele propoziţii:

a) Histogramele .... se trasează mult mai încet decît histogramele trasate prin metoda ....

b) Microcalculatoarele .... , ... , ... sînt prevăzute cu opţiune grafică.

c) În BASIC-aMIC instrucţiunile **WINDOW** şi .... sînt executate implicit.

d) Instrucțiunea **MOVE** este folosită pentru .... în timp ce instrucțiunea **RMOVE** ....

e) Instrucțiunea **DRAW** este folosită pentru ...., ..., ...

f) Instrucțiunea **RDRAW** este folosită pentru ....

g) Instrucțiunea prin care utilizatorul specifică limitele „spațiului utilizator" în care sînt cuprinse datele de reprezentat este ....

h) Instrucțiunea **VIEWPORT** specifică ....

i) Instrucțiunile ...., ..., ..., ..., formează setul minim de instrucțiuni grafice în limbajul BASIC-AMSTRAD.

j) Pentru programarea culorilor în BASIC TIM S se utilizează ...., ...

k) Instrucțiunea **PLOT** nu aparține limbajului BASIC implementat pe calculatoarele ...., ..., ..., ..., ...

☐ Scrieți cîte un program BASIC pentru fiecare din problemele de mai jos:

a) Să se genereze un con cu vîrful în punctul de coordonate (a, b).

b) Să se genereze n dreptunghiuri incluse unul într-altul, primul dreptunghi trasat avînd dimensiunile cele mai mari.

c) Să se genereze n triunghiuri isoscele cu laturile paralele și incluse unul într-altul.

d) Să se scrie un program BASIC HC-85, TIM S, SPECTRUM care selectează aleator codul unui caracter și-l afișează pe ecran într-o poziție aleatoare.

e) Să se deseneze un triunghi definit prin coordonate generate aleator.

f) Să se genereze un cerc prin mai multe metode.

g) Să se genereze un octogon.

h) Să se genereze mai multe poligoane concentrice.

i) Să se genereze mai multe figuri stelate.

j) Se consideră o placă plană a cărei suprafață poate fi împărțită în dreptunghiuri (pătrate) și triunghiuri. Să se scrie un program BASIC-80 care:

- trasează conturul plăcii plane;
- determină poziția centrului de greutate al plăcii.

☐ **Să se traseze în 2D și 3D** histograma corespunzătoare cantităților lunare de jucării fabricate de Întreprinderea URSULEȚUL.

☐ **Să se reprezinte procentele** corespunzătoare consumurilor trimestriale ale unui articol cu ajutorul unei histograme (linie) în 3D.

SOLUŢIA TEMEI 12

☐ Nu răspundem.

☐ Nu răspundem.

☐ **Programele BASIC sînt:**

a1) *Lista programului BASIC–AMSTRAD*

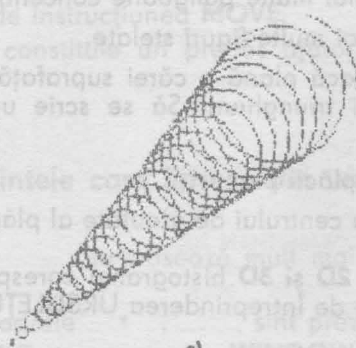
```
10 CLS
20 READ a, b
24 DATA 100, 100
26 LOCATE 1,1 : PRINT "pasul este"; : INPUT h
27 CLS
28 n=400
30 FOR r=1 TO n STEP 5
40    MOVE a+r, b
50    FOR i=0 TO 2*PI STEP PI/60
60       DRAW a+r*COS (i), b+r*SIN (i)
70    NEXT i
72    a=a+h
75    b=b+h
80 NEXT r
90 GOTO 90
```



a)



b)



c)

Fig. 12.33. Con: *a)* pasul 2; *b)* pasul 3; *c)* pasul 4.

a2) *Lista programului BASIC HC-85, TIM S, SPECTRUM*

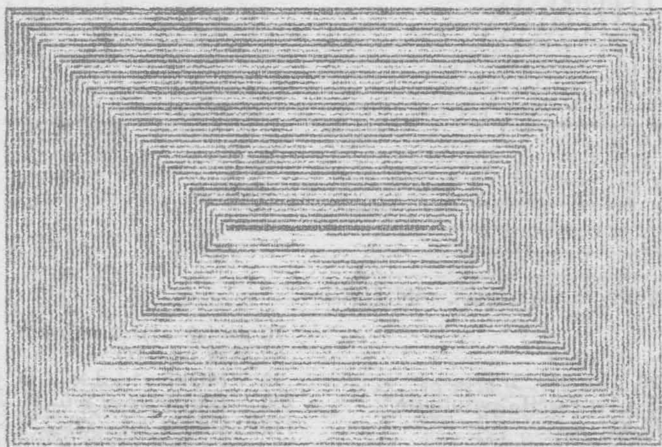
```
10 CLS
20 READ a, b
24 DATA 50, 50
26 PRINT AT 1,1; "pasul este"; : INPUT h
27 CLS
28 LET n=50
30 FOR r=1 TO n
40    CIRCLE a, b, r
72    LET a=a+h
75    LET b=b+h
80 NEXT r
90 GOTO 90
```

b1) *Lista programului BASIC–AMSTRAD*

```
10 CLS
20 n=1000
30 READ x, y, 1, d
40 DATA 20, 20, 360, 600
50 LOCATE 1,1 : PRINT "pasul
   este"; : INPUT h
60 CLS
70 FOR i=1 TO n
80    MOVE x+h * (i-1),
      y+h * (i-1)
90    l1=l-2 * h * (i-1)
100   IF l1<0 THEN 170
110   DRAW x+h * (i-1),
      y+h * (i-1)+l1
120   d1=d-2 * h * (i-1)
130   DRAW x+h * (i-1)+d1,
      y+h *(i-1)+l1
140   DRAW x+h * (i-1)+d1,
      y+h * (i-1)
150   DRAW x+h * (i-1),
      y+h * (i-1)
160 NEXT i
170 GOTO 170
```

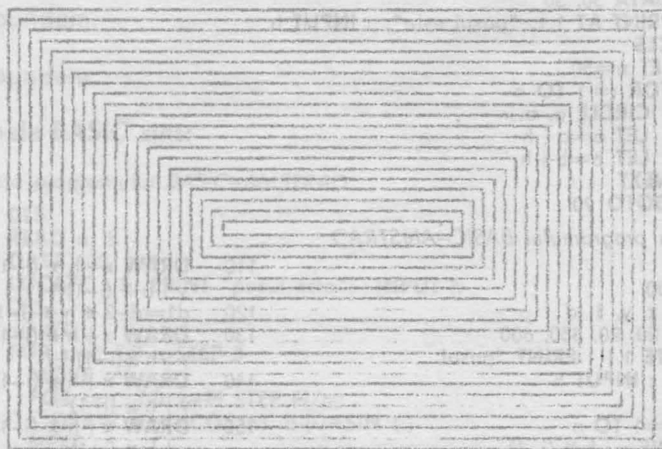
b2) *Lista programului BASIC HC-85, TIM S, SPECTRUM*

```
10 CLS
20 LET n=1000
30 READ x, y, l, d
40 DATA 5, 5, 245, 165
50 PRINT AT 1,1; "pasul este": INPUT h
60 CLS
70 FOR i=1 TO n
80    PLOT x+h * (i-1), y+h * (i-1)
90    LET l1=l-2 * h * (i-1)
95    LET d1=d-2 * h * (i-1)
97    IF d1<0 THEN GO TO 170
100   IF l1<0 THEN GO TO 170
110   DRAW l1, 0
120   LET d1=d-2 * h * (i-1)
125   IF d1<0 THEN GO TO 170
130   DRAW 0, d1
140   DRAW -l1, 0
150   DRAW 0, -d1
160 NEXT i
170 GO TO 170
```

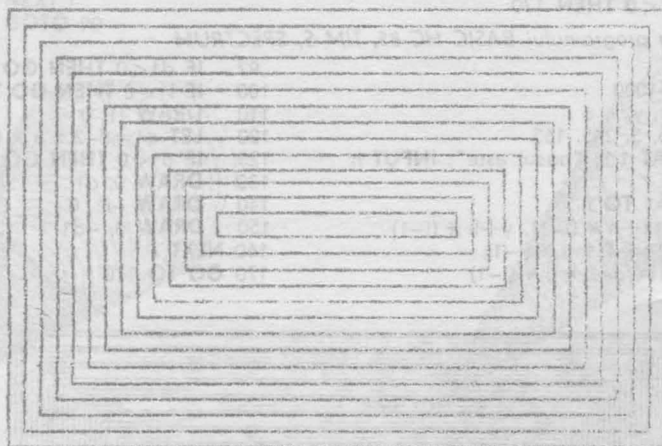


a)

Fig. 12.34. Dreptunghiuri imbricate *(a, b, c)*.



b)



c)

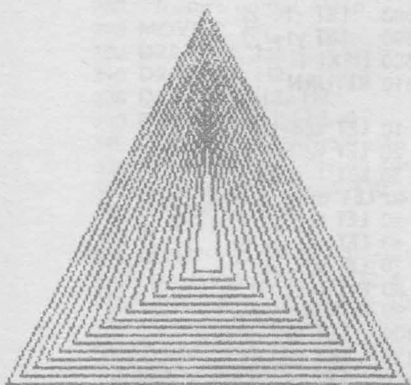
Fig. 12.34.

c1) *Lista programului BASIC–AMSTRAD*

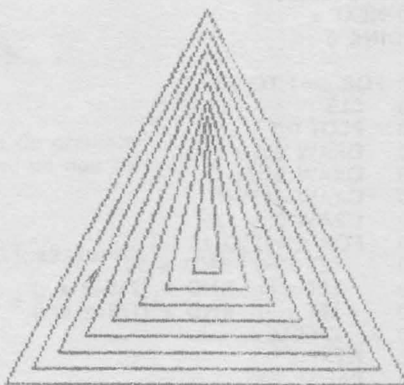
```
10 CLS
20 READ y, x1, x2, h
24 DATA 20, 100, 490, 375
26 LOCATE 1,1 : PRINT "pasul este";
   : INPUT h1
27 CLS
40 t=(2 * h)/(x2–x1)
50 t2=(–1+SQR (t ↑ 2+1))/t
60 d=h1/t2
70 n=(x2–x1)/(2 * d)+1
80 FOR i=1 TO n
90    MOVE x1+d * (i–1), y+h1 * (i–1)
100   DRAW x2–d * (i–1), y+h1 * (i–1)
110   DRAW (x1+x2)/2, y+h1 * (i–1)+
      t * ((x2–x1)/2–d * (i–1))
120   DRAW x1+d * (i–1), y+h1 * (i–1)
130 NEXT i
140 GOTO 140
```

c2) *Lista programului BASIC HC-85, TIM S, SPECTRUM*

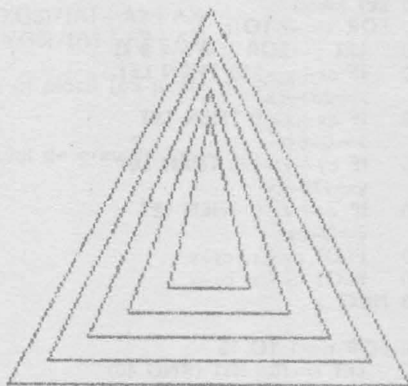
```
10 CLS
20 READ y, x1, x2, h
25 DATA 10, 50, 200, 140
26 PRINT AT 1,1; "pasul este"; :
   INPUT h1
27 CLS
40 LET t=(2 * h)/(x2-x1)
50 LET t2=(-1+SQR (t ↑ 2+1))/t
60 LET d=h1/t2
70 LET n=(x2-x1)/(2 * d)+1
80 FOR i=1 TO n
90    PLOT x1+d * (i-1), y+h1 * (i-1)
95    LET l=x2-x1-2 * d * (i-1)
100   DRAW l, 0
105   LET a=h1 * (i-1)+t * ((x2-x1)/2
      -d * (i-1))
110   DRAW -l/2, a
115   DRAW -l/2, -a
130 NEXT i
```



a)



b)



c)

Fig. 12.35. Triunghiuri imbricate *(a, b, c)*.

d)

```
10 FOR n=1 TO 100
20    LET r=INT (RND * 22)
30    LET c=INT (RND * 32)
40    INK INT (RND * 7)
50    LET s=96+INT (RND * 48)
60    PRINT AT r, c : CHR$ s;
70 NEXT n
```

e)

```
10 FOR s=1 TO 20
20    BORDER INT (RND * 8)
30    LET p=INT (RND * 8)
40    PAPER p : CLS
50    FOR n=1 TO 15
60       LET x1=INT (RND * 255)
70       LET y1=INT (RND * 175)
```

```
 80    LET x2=INT (RND * 255)
 90    LET y2=INT (RND * 175)
100    LET x3=INT (RND * 255)
110    LET y3=INT (RND * 175)
120    LET c=INT (RND * 8)
130    IF c=p THEN GO TO 80
140    INK c
150    PLOT x1, y1
160    DRAW x2-x1, y2-y1
170    DRAW x3-y2, y3-y2
180    DRAW x1-x3, y1-y3
190 NEXT n
200 PAUSE 200
210 NEXT s
220 INK 0
```

f1)

```
 10 FOR s=1 TO 10
 20    CLS
 30    PLOT 0,0
 40    DRAW 255,0
 50    DRAW 0,175
 60    DRAW -255,0
 70    DRAW 0, - 175
 80    FOR n=1 TO 15
 90       LET r=5+RND * 50
100       LET x=r+RND * (255-2 * r)
110       LET y=r+RND * (175-2 * r)
120       CIRCLE x, y, r
130    NEXT n
140    PAUSE 200
150 NEXT s
160 STOP
```

f2)

```
 10 LET cx=128
 20 LET cy=96
 30 LET r=50
 40 FOR x=-r TO r
 50    LET y=SQR (r * r-x * x)
 60    IF cx+x>255 THEN LET
       x=255-cx
 70    IF cx+x<0 THEN LET
       x=0-cx
 80    IF cy+y>175 THEN LET
       y=175-cy
 90    IF cy+y<0 THEN LET
       y=0-cy
100    PLOT cy+x, cy-y
110    PLOT cx+x, cy-y
120 NEXT x
```

f3)

```
 10 FOR n=1 TO 15
 20    LET r=10+INT (RND 40)
 30    LET x=r+INT (RND+
       (255-2 * r))
 40    LET y=r+INT (RND
       (175-2 * r))
 50    GOSUB 500
 60 NEXT n
 70 STOP
500 LET th=2 * PI/r
510 LET x1=r
520 LET y1=0
530 PLOT x+x1, y+y1
540 FOR i=1 TO r
550    LET x2=x1 * COS th-
       y1 * SIN th
560 LET y2=x1 * SIN th+y1 *
       COS th
570    DRAW x2-x1, y2-y1
580    LET x1=x2
590    LET y1=y2
600 NEXT i
610 RETURN
```

g)

```
 10 LET xc=128
 20 LET yc=88
 30 LET r=50
 40 LET dt=2 * PI/8
 50 LET th=0
 60 LET x1=xc+r
 70 LET y1=yc
 80 PLOT x1, y1
 90 FOR n=1 TO 8
100    LET th=dt * n
110    LET x2=xc+r * COS th
120    LET y2=yc+r * SIN th
130    DRAW x2-x1, y2-y1
140    LET x1=x2
150    LET y1=y2
160 NEXT n
```

h)

```
 10 LET r=12
 20 LET xc=128
 30 LET yc=88
 40 FOR k=3 TO 9
 50    LET ns=k
 60    LET DT=2 * PI/ns
 70    LET x1=xc+r
 80    LET y1=yc
 90    PLOT x1, y1
100    FOR n=1 TO ns
110       LET th=n * dt
120       LET x2=xc+r * COS th
130       LET y2=yc+r * SIN th
140       DRAW x2-x1, y2-y1
150       LET x1=x2
160       LET y1=y2
170    NEXT n
180    LET r=r+12
190 NEXT k
```

j) *Lista programului BASIC-80*

```
100 'Inițializare grafică'
110 PRINT CHR$(27)+"I"+CHR$(27)+"2"
120 'Introducerea datelor și calculul centrului de greutate'
130 GOSUB 390
140 VIEWPORT 0, 133, 0, 100 ' stabilire fereastră fizică '
```

```
150 WINDOW -30, 1330, -20, 1000 ' stabilire fereastră logică '
160 ' Trasarea axelor '
170 ROTATE 0
180 MOVE 0, 0
190 DRAW 1300, 0
200 RDRAW -30, 30
210 RMOVE 0, -60
220 RDRAW 30, 30
230 MOVE 0, 0
240 RDRAW 0, 980
250 RDRAW -30, -30
260 RMOVE 60, 0
270 RDRAW -30, 30
280 ' Trasare contur placă '
290 MOVE 0, H1
300 DRAW L1, H1
310 DRAW L1, H2
320 DRAW L1+L2, H2
330 DRAW L1+L2+L3, 0
340 GOSUB 540 ' Vizualizarea centrului de greutate '
350 ' Testarea condiției de continuare cu un nou calcul '
360 IF DN$="D" THEN 110
370 GOSUB 790
380 END
390 INPUT " Date de intrare (H1, H2, L1, L2, L3) "; H1, H2, L1, L2, L3
400 H1=H1 * 10 : H2=H2 * 10
410 L1=L1 * 10 : L2=L2 * 10 : L3=L3 * 10
420 XG1=L1/2
430 YG1=H1/2
440 XG2=L1+L2/2
450 YG2=H2/2
460 XG3=L1+L2+L3/3
470 YG3=H2/3
480 A1=L1 * H1
490 A2=L2 * H2
500 A3=L3 * H2/2
510 XG=(A1 * XG1+A2 * XG2+A3 * XG3)/(A1+A2+A3)
520 YG=(A1 * YG1+A2 * YG2+A3 * YG3)/(A1+A2+A3)
530 RETURN
540 ' Localizarea centrului de greutate al plăcii (un punct pe ecran) '
550 MOVE INT(XG), INT(YG)
560 RDRAW 0, 0
570 ' Trasarea unui cerc în jurul centrului de greutate găsit '
580 N=30
590 IU=8 * ATN(1)/N
600 U=0
610 FOR I=1 TO N
620   MOVE INT(XG), INT(YG)
630   ROTATE U
640   RMOVE 30, 0
650   RDRAW 0, 3
660   U=U+IU
670 NEXT I
680 XG=XG/10 : YG=YG/10
690 PRINT CHR$(14) ' Video invers '
700 ' Tipărirea pe ecran a coordonatelor centrului de greutate '
710 PRINT CHR$(27)+"1"+CHR$(35)+CHR$(54); "XG="; XG; "YG="; YG; " "
720 PRINT CHR$(15) ' Revenire în video normal '
730 PRINT CHR$(27)+"1"+CHR$(45)+CHR$(33);
740 GOSUB 810 'Tipărirea la imprimantă a rezultatelor '
750 PRINT "Doriți alt calcul (D/N)";
760 DN$=INPUT$(1)
770 RETURN
```

```
780 ' Revenire în modul de lucru defilare '
790 PRINT CHR$(27)+"2"
800 RETURN
810 LPRINT " Determinarea centrului de greutate al unei plăci "
820 LPRINT " avînd dimensiunile : "
830 LPRINT " H1= " ; H1/10
840 LPRINT " H2=" ; H2/10
850 LPRINT " L1=" ; L1/10
860 LPRINT " L2=" ; L2/10
870 LPRINT " L3=" ; L3/10
880 LPRINT " Centrul de greutate are coordonatele : "
890 LPRINT " XG=" ; XG ; " YG=" ; YG
900 RETURN
RUN
```

```
Determinarea centrului de greutate al unei plăci
avînd dimensiunile:

        H1=53
        H2=40
        L1=25
        L2=46
        L3=36

Centrul de greutate are coordonatele:

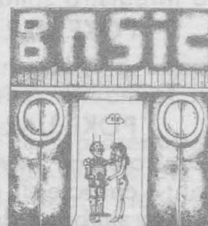
        XG=42.379 YG=20.9813
```

☐ Nu răspundem.

☐ Nu răspundem.

CONVERSAȚIA 13

Simularea livrărilor de lichide din trei rezervoare cu ajutorul imaginilor animate. Simboluri grafice definite de utilizator. Sprite-uri COMMODORE. Simboluri grafice în mișcare. JOCURI, APLICAȚII și TESTE pentru cititor



□ Caractere definite de utilizator pe calculatoarele HC-85, TIM S, SPECTRUM și AMSTRAD

Pînă acum am desenat cu calculatorul folosindu-ne numai de setul de caractere standard sau de caracterele mozaic. Ce facem însă dacă dorim să reproducem o literă grecească ori un alt simbol și mai complicat? O soluție salvatoare ar fi utilizarea pe cît posibil a instrucțiunilor **PLOT** și **DRAW** dar, nu întotdeauna metoda este și practică. Ceea ce dorim să vă prezentăm în cadrul acestei conversații este o metodă pentru generarea caracterelor (simbolurilor) nestandard care pot fi tipărite la fel ca și simbolurile obișnuite.

Și ceea ce este minunat este faptul că atît HC-85, TIM S, SPECTRUM cît și calculatorul personal AMSTRAD dispun de o astfel de facilitate.

Metodă pentru programarea simbolurilor grafice definite de utilizator în BASIC HC-85, TIM S și SPECTRUM

1. Căutarea modelului necesar.
2. Desenarea simbolului dorit pe o grilă de 8×8 pixeli (puncte reprezentate prin pătrate). Pătratele se vor înnegri pentru a da forma simbolului dorit.
3. Definirea valorilor binare cu ajutorul funcției **BIN** corespunzătoare simbolului definit. „Zero" semnifică un pătrat alb iar „unu" semnifică un pătrat „negru".
4. Introducerea simbolului în memorie cu ajutorul instrucțiunii **POKE** al cărei format general este:

Format general
POKE ⟨m⟩, ⟨n⟩

unde ⟨m⟩ reprezintă adresa de memorie;
⟨n⟩ valoarea pe care dorim s-o înscriem.

Observație. Această înscriere nu se realizează prin intermediul procedeelor BASIC obișnuite (0⩽m⩽65535; 0⩽n⩽255). Opusul lui **POKE** este **PEEK** care citește din memorie octetul de la adresa <m>).

Din fericire pentru noi, nu este necesar să știm adresa de memorie (⟨m⟩) la care dorim să înscriem simbolul pe care l-am definit, deoarece calculatorul se poate „descurca" și singur analizînd funcția **USR** "M", unde M reprezintă caracterul căruia îi va fi afectat simbolul definit de noi. Pentru a înregistra simbolul, vom scrie:

POKE USR "M"+i, n

unde n reprezintă valoarea din linia ⟨i⟩ de puncte. Se mai poate utiliza și forma de scriere

POKE USR "M"+i, **BIN** (linia i, 8 biți).

De notat că în mod grafic, ori de cîte ori vom acționa tasta [M] (M reprezintă caracterul căruia i s-a atribuit simbolul definit de noi) se va afișa pe ecran simbolul definit.

Observație. În BASIC-AMSTRAD pentru definirea caracterelor utilizator se folosește instrucțiunea **SYMBOL** (v. vol. 2, pag. 57).

Aplicații

1. Folosind tehnica generării simbolurilor grafice definite de utilizator programați tasta [A] pentru reprezentarea unui pătrat și apoi a unui elefant.

a)
```
10 FOR J=0 TO 7
20    READ B
30    POKE USR "A"+J, B
40 NEXT J
50 PRINT A
60 DATA BIN 00000000
65 DATA BIN 00001000
70 DATA BIN 00010100
75 DATA BIN 00100010
80 DATA BIN 01000001
85 DATA BIN 00100010
90 DATA BIN 00010100
95 DATA BIN 00001000 .
100 STOP
```

b) Se înlocuiesc liniile 60–95 cu
```
60 DATA BIN 00110000
65 DATA BIN 01110000
70 DATA BIN 01111110
75 DATA BIN 01011111
80 DATA BIN 10011111
85 DATA BIN 00010010
90 DATA BIN 00010010
95 DATA BIN 00000000
```

2. Introduceți și executați următoarele programe:

a)
```
10 LET x=0
20 PRINT AT 5, x; "■"
30 PRINT AT 5, x; " "
40 LET x=x+1
50 GOTO 20
```

b)
```
10 LET x=0
20 PRINT AT 5, x; "■"
30 FOR i=1 TO 10
40 NEXT i
50 PRINT AT 5, x; " "
60 LET x=x+1
70 IF x>31 THEN LET x=0
80 GO TO 20
```

c)
```
10 LET x=0
20 LET p=x
30 LET x=x+1
40 IF x>31 THEN LET x=0
50 PRINT AT 5, p; " "; AT 5, x; "■"
60 GO TO 20
```

d)
```
10 FOR x=0 TO 30
20 PRINT AT 5, x; "■"; AT 5,31;"
   "AND x=0
30 NEXT x
40 GO TO 10
```

e)
```
10 POKE USR "A", BIN 00011100
20 POKE USR "A"+1, BIN 00011100
30 POKE USR "A"+2, BIN 00001000
40 POKE USR "A"+3, BIN 00001000
50 POKE USR "A"+4, BIN 00001000
60 POKE USR "A"+5, BIN 00011100
70 POKE USR "A"+6, BIN 00100010
80 POKE USR "A"+7, BIN 00100010
90 FOR i=1 TO 20
100    LET r=INT (RND*20)
110    LET c=INT (RND*30)
120    PRINT AT r, c; CHR$ (144);
130 NEXT i
```

f)
```
10 LET y=0
20 LET x=0
30 PRINT AT y, x; "■"
40 IF INKEY$="5" THEN LET x=x-1
50 IF x<=0 THEN LET x=0
60 IF INKEY$="6" THEN LET y=y+1
70 IF y>=21 THEN LET y=21
80 IF INKEY$="7" THEN LET y=y-1
90 IF y<=0 THEN LET y=0
100 IF INKEY$="8" THEN LET x=x+1
110 IF x>=31 THEN LET x=31
120 GO TO 30
```

□ Generarea sprite-urilor pe calculatorul COMMODORE

Ce este un sprite? Un sprite (spiriduș) este o formă de reprezentare grafică de înaltă rezoluție pe calculatorul personal COMMODORE ce presupune existența unei grile cu 24×21 puncte (504 puncte în total). Fiecare linie a sprite-ului este divizată în trei grupe a cîte 8 puncte fiecare. Prima dumneavoastră sarcină în desenarea unui sprite este de a decide care din

aceste puncte doriți să se „aprindă" pentru a forma o figură sprite. Sprite-urile pot fi realizate și color (patru culori)! Îndată ce-ați umbrit sprite-ul, trebuie să aduceți figura la o formă pe care calculatorul să o poată înțelege. Fiecare din cele trei numere binare (1 corespunde pătratului înnegrit) urmează a fi convertit în echivalentul lui zecimal. Procedura se repetă pentru toate cele 21 de linii ale sprite-ului.

SPRITE-uri originale

Presupunînd că nu aveți înclinații artistice, de unde vă puteți inspira pentru a desena figuri sprite? Literele alfabetului, cifrele construite din mai multe linii drepte, pot constitui o sursă. Priviți de asemenea la reclamele

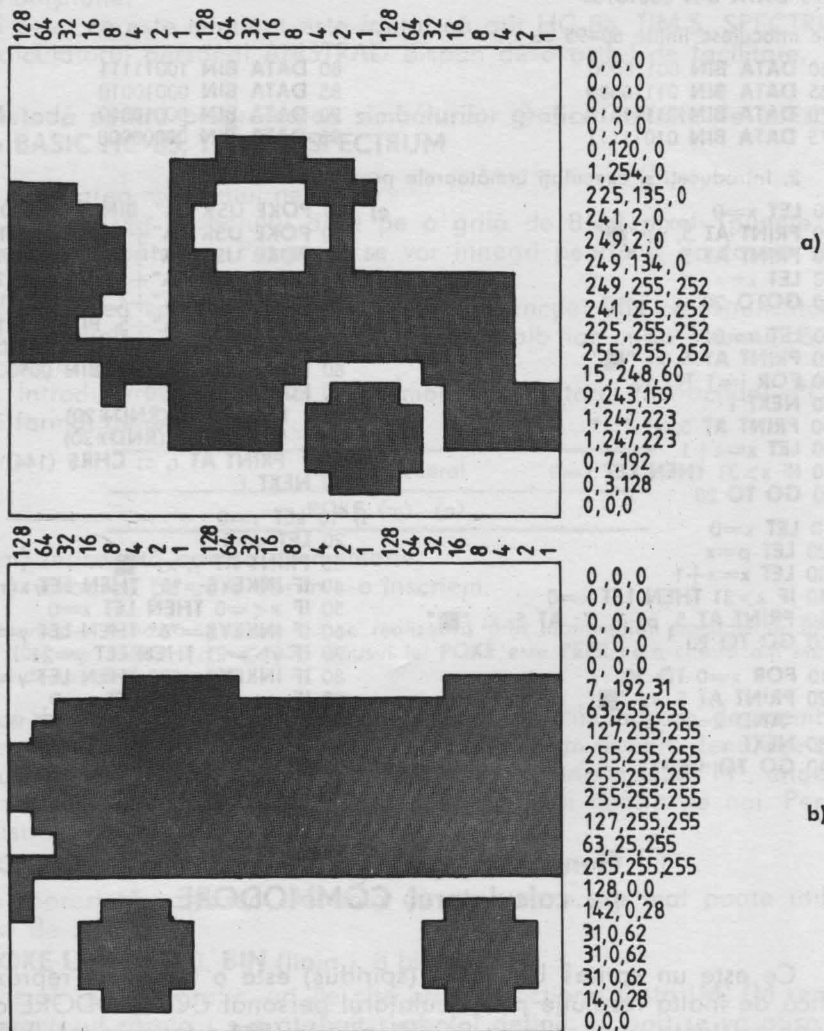


Fig. 13.1. Generarea unei cisterne cu petrol utilizînd tehnica sprite-urilor.

de pe marile străzi luminate! Nu neglijați nici cusăturile artizanale, șabloanele ori vederile colorate pe care de regulă . . . le primiți!

Dacă preferați să așteptați (nefiind grăbiți să desenați propriile dvs. sprite-uri) ne permitem să vă prezentăm două sprite-uri originale reprezentînd o cisternă cu petrol (v. fig. 13.1).

Stocarea sprite-urilor

Figurile sprite (putem lucra cu 8 sprite-uri, numerotate de la 0–7) se definesc cu instrucțiunile **DATA** care conțin 63 de valori. Stocarea unui sprite în memorie implică două faze distincte: a) precizarea adresei de memorare; b) introducerea propriu-zisă a valorilor sprite-ului la adresa indicată.

Indicarea adresei de memorare

Fiecare din cele 8 sprite-uri cu care putem lucra în BASIC-COMMODORE are asociat un pointer. SPRITE POINTER-ul vi-l puteți imagina ca pe cuprinsul unei cărți care ne indică de fiecare dată unde începe fiecare capitol al lucrării. SPRITE POINTER-ul transmite programului adresa de încărcare în memorie a valorilor declarate în **DATA** pentru fiecare sprite. Astfel, sprite-ul zero are pointer-ul 2040, . . . , sprite-ul 7 are pointer-ul 2047.

Introducerea valorilor

După cum știți nu toată memoria calculatorului COMMODORE este la dispoziția dvs.! Amintiți-vă că BASIC-ul cît și anumite componente ale sistemului de operare sînt rezidente!

Prin stabilirea SPRITE POINTER-ului pentru sprite-ul #0 la adresa 192, de exemplu, valorile datelor pentru acest sprite trebuie citite în memorie la adresele (192*64)+0 pînă la (192*64)+62. Această operație poate fi ușor realizată utilizînd o buclă **FOR-NEXT.**

```
10 V=53248
20 FOR T=0 TO 62
30   READ A
40   POKE 192*64+T, A
50 NEXT T
```

Deoarece toate cele opt sprite-uri pot fi diferite, avînd propriile lor valori **DATA,** rezultă că fiecare sprite va trebui să fie localizat distinct.

O simplă buclă **FOR-NEXT** poate realiza acest lucru.

```
10 FOR T=0 TO 7
20   POKE 2040+T,192+T
30 NEXT T
```

Acum, cînd avem precizată aria din memorie pentru valorile blocului de date **DATA** urmează doar ca aceste valori să fie citite.

```
10 FOR T=0 TO 62
20   READ A
30   POKE 192*64+T, A
40 NEXT T
```

Secvenţa de instrucţiuni BASIC-COMMODORE de mai sus va plasa cele 63 de valori (definite în **DATA**) ale sprite-ului #0 în locaţiile de memorie (192 * 64)+0 pînă la (192 * 64)+62. Vom proceda analog pentru celelalte şapte sprite-uri modificînd adresa zonei de memorie.

```
10 FOR T=0 TO 62 : READ A : POKE 193*64+T, A : NEXT T
20 FOR T=0 TO 62 : READ A : POKE 194*64+T, A : NEXT T
30 FOR T=0 TO 62 : READ A : POKE 195*64+T, A : NEXT T
40 FOR T=0 TO 62 : READ A : POKE 196*64=T, A : NEXT T
50 FOR T=0 TO 62 : READ A : POKE 197*64+T, A : NEXT T
60 FOR T=0 TO 62 : READ A : POKE 198*64=T, A : NEXT T
70 FOR T=0 TO 62 : READ A : POKE 199*64+T, A : NEXT T
```

Din acest moment puteţi privi liniştit orice sprite. Dacă doriţi să citiţi fiecare din cele opt sprite-uri în zone de memorie succesive, utilizaţi secvenţa de instrucţiuni:

```
10 FOR B=0 TO 7
20   FOR T=0 TO 62
30     READ A
40     POKE (192+B)*64+T, A
50   NEXT T
60 NEXT B
```

Observaţie. Nu întotdeauna aveţi nevoie de opt sprite-uri diferite în programele dvs. Să presupunem că doriţi opt sprite-uri identice. Procedaţi astfel:

```
10 POKE 2040, 192
20 POKE 2041, 192
30 POKE 2042, 192
40 POKE 2043, 192
50 POKE 2044, 192
60 POKE 2045, 192
70 POKE 2046, 192
80 POKE 2047, 192
```

Şi-acum, vă propunem, un program original pentru generarea unei cisterne pornind de la cele două sprite-uri pe care vi le-am prezentat la început.

```
 10 Q=53248
 20 FOR I=0 TO 62
 30     READ A
 40     POKE 255*64+I, A
 50 NEXT I
 60 POKE 2040, 255
 65 FOR I=0 TO 62
 70     READ A
 75     POKE 254*64+I, A
 80 NEXT I
 85 POKE 2041, 254
 90 POKE Q+21,3
 95 POKE Q+39,1 : POKE Q+40,1
100 POKE Q+130 : POKE Q+1, 130.
110 POKE Q+2, 178 : POKE Q+3, 130
120 POKE Q+23,3 : POKE Q+29,3
130 GET A$ : IF A$=" " THEN 130
140 POKE Q+21,0
160 DATA 0, 0, 0, 0, 0, 0, 0, 0
170 DATA 0, 0, 0, 0, 0, 0, 0, 7
180 DATA 192, 31, 63, 255, 255, 127, 255, 255
190 DATA 255, 255, 255, 255, 255, 255 255, 255
200 DATA 255, 127, 255, 255, 63, 255, 255, 127
210 DATA 255, 255, 7, 1, 192, 3, 57, 128
220 DATA 1, 125, 0, 1, 109, 0, 0, 124
```

```
230 DATA 0, 0, 36, 0, 0, 0
240 DATA 0, 0, 0, 0, 0, 0, 0, 0
250 DATA 0, 0, 0, 0, 0, 120, 0, 1
260 DATA 254, 0, 225, 135, 0, 241, 2, 0
270 DATA 249, 2, 0, 249, 134, 0, 249, 255
280 DATA 248, 241, 255, 248, 225, 225, 248, 255, 255,
    248, 255, 248, 56, 255, 243, 152
290 DATA 1, 247, 220, 1, 246, 220, 0, 7, 192, 0, 3, 128, 0, 0, 0
```

Aplicație

1. Includeți în programele dvs. următoarele sprite-uri reprezentînd:

a) O ceașcă de cafea.

```
10000 REM CEAȘCA DE CAFEA
10010 DATA 8
10020 DATA 66, 0, 4, 33, 0
10030 DATA 2, 16, 128, 4, 33
10040 DATA 0, 8, 66, 0, 0
10050 DATA 0, 0, 255, 255, 192
10060 DATA 255, 255, 192, 245, 85
10070 DATA 252, 245, 85, 254, 245
10080 DATA 85, 199, 245, 85, 195
10090 DATA 245, 85, 195, 245, 85
10100 DATA 195, 245, 85, 199, 245
10110 DATA 85, 206, 245, 85, 252
10120 DATA 245, 85, 248, 245, 85
10130 DATA 240, 245, 85, 192, 255
10140 DATA 255, 192
```

b) Un lup.

```
10280 DATA 0, 0, 0, 0, 0
10290 DATA 0, 0, 0, 48, 1
10300 DATA 192, 112, 0, 240, 240
10310 DATA 0, 121, 224, 0, 61
10320 DATA 192, 0, 31, 192, 0
10330 DATA 15, 224, 0, 63, 224
10340 DATA 0, 255, 240, 127, 199
10350 DATA 248, 255, 255, 252, 255
10360 DATA 255, 254, 255, 255, 254
10370 DATA 128, 255, 254, 42, 255
10380 DATA 254, 127, 255, 252, 1
10390 DATA 255, 248, 1, 255, 240
10400 DATA 1, 255, 240
```

2. Introduceți și executați următorul program care produce mișcarea unui fluture.

```
 15 V=53248
 30 PRINT CHR$ (147) : POKE V+21,0
 40 POKE 53281,7 : POKE 53280,5
 90 POKE 2040, 192
110 FOR M=0 TO 162
120   READ A
130   POKE 192*64+M, A
140 NEXT M
150   POKE V+39,6
165   POKE V+0, 100
170   FOR Y=1 TO 255
180   POKE V+1, Y
185   POKE V+21,1
190   NEXT Y
195 GO TO 170
10000 DATA 3, 0, 192, 57, 129
10010 DATA 156, 124, 195, 62, 254
10020 DATA 102, 127, 255, 36, 255
10030 DATA 255, 155, 255, 255, 219
10040 DATA 255, 255, 255, 255, 255
10050 DATA 255, 255, 255, 255, 255
10060 DATA 255, 255, 255, 255, 255
10070 DATA 255, 255, 255, 255, 255
10080 DATA 255, 255, 255, 255, 255
10090 DATA 255, 255, 255, 255, 255
10100 DATA 255, 127, 219, 254, 63
10110 DATA 153, 252, 31, 24, 248
10120 DATA 14, 24, 112
```

3. Introduceți și executați următorul program:

```
 5 REM SIMULAREA LUI DRAW TO
10 INPUT "INTRODUCEȚI X1"; X1
20 INPUT "INTRODUCEȚI Y1"; Y1
30 INPUT "INTRODUCEȚI X2"; X2
40 INPUT "INTRODUCEȚI Y2"; Y2
42 POKE 53265, 59
44 POKE 53272,25
45 FOR V=1024 TO 2023
46 POKE V, 16
47 NEXT V
48 FOR T=8192 TO 16383
49 POKE T, 0
50 NEXT T
55 DX=X2-X1 : DY=Y2-Y1
57 IF ABS(DY)<ABS(DX) THEN 64
58 FOR YL=Y1 TO Y2 STEP SGN(DY)
59 X=DX/DY*YL+X1
60 Y=YL
61 GOSUB 200
62 NEXT YL
63 GOTO 300
64 FOR XL=X1 TO X2 STEP SGN (DX)
65 X=XL
66 Y=DY/DX*XL+Y1
67 GOSUB 200
68 NEXT XL
69 GOTO 300
200 YP=INT (Y/8)
210 XP=INT (X/8)
220 A1=(YP*40+XP)*8
230 AY=Y-8*YP+A1
240 AA=8192
250 R=X-8*XP
260 M=2 ↑ (7-R)
```

```
270 I=PEEK(AY+AA)
280 POKE AY+AA, I OR M
290 RETURN
```

```
300 FOR T=1 TO 5000 : NEXT T
310 POKE 53265,27 : POKE 53272,21
320 END
```

JOC PE COMMODORE

```
100 REM PARABOLA
110 PRINT CHR$ (147)
120 X0=0 : Y0=0
130 PRINT CHR$ (147) : INPUT "INTRODUCEŢI
    UNGHIUL"; W
140 INPUT "INTRODUCEŢI VO" ; VO
150 W=W*2*3.1416/360
160 VX=VO*COS(W)
170 VY=VO*SIN(W)
180 G=9.81
190 T=0 : H=0 : L=0
200 Y=-G/2*T*T+VY*T
210 IF Y/0 THEN FOR D=1 TO 500 :
    NEXT D : GOTO 130
220 X=VX*T
230 LOC=1024+40*INT (24-Y/50)+INT
    (X/50)
240 COL=55296+40*INT (24-Y/50)+INT
    (X/50)
250 IF LOC<1024 THEN T=T+1 : GOTO
    200
260 POKE LOC, 81 : POKE COL, 4
270 L=X : H=Y
280 T=T+1
290 GOTO 200
```

□ Programarea imaginilor animate pe calculatorul aMIC

Deplasarea unui caracter semigrafic

Teoria privind mişcarea unei imagini grafice precizează următoarele posibilităţi: a) se trasează graficul; b) se produce o întîrziere; c) se şterge graficul; d) se trasează un nou grafic; d) se reia ciclul descris.

Efectul de mişcare a unui caracter grafic poate fi obţinut prin suprimarea iluminării acestuia nu cu mult timp înainte de apariţia unui caracter identic în poziţia imediat următoare sau prin plasarea în aceeaşi poziţie a unui caracter grafic urmat sau nu de un blanc. Pentru definirea poziţiei caracterelor se utilizează variabilele (de stare).

Să privim exemplele care urmează ca pe nişte aplicaţii „brutale" ale teoriei animaţiei. Treziţi aMIC-ul dvs!

```
10 INIT
15 FOR J=16 TO 25
20   PRINT AT (16, J); "■"
25   PRINT AT (16, J); " "
30 NEXT J
40 END
```

Aţi avut o impresie de deplasare paralelă cu axa Ox, a caracterului "■"? Încercaţi următoarea secvenţă, mult mai rapidă.

```
10 INIT
20 FOR J=15 TO 24
25    PRINT AT (16, J); " ■"
30 NEXT J
40 END
```

După cum aţi putut constata cele două secvenţe sînt identice.

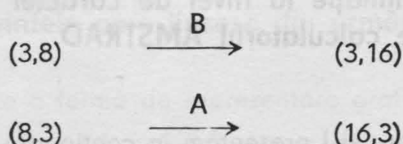
Obţineţi acum acelaşi efect de deplasare a caracterului "■" pe verticală!

```
10 INIT
20 FOR I=16 TO 25
25     PRINT AT (I, 15); "■"
30     PRINT AT (I, 15); " "
40 NEXT I
```

Aplicație. Să se simuleze deplasarea caracterului semigrafic "■" pe drumul cel mai scurt dintre punctele A (1, 2) și B (16, 17).

Deplasarea simultană a două caractere

Să considerăm cazul simulării deplasării a două caractere A şi B legate între ele printr-o funcţie.



După cum constataţi, funcţia de legătură este evidentă (punctele sînt simetrice faţă de diagonală).

```
5 INIT
10 FOR I=8 TO 16
15    PRINT AT (3, I); "B"
20    PRINT AT (3, I); " "
25    PRINT AT (I, 3); "A"
30    PRINT AT (I, 3); " "
40 NEXT I
50 END
```

Observație. Impresia de deplasare se datorează rapidității mișcării (instrucțiunile se execută una după alta).

Deplasarea unei figuri mai complicate

Să abordăm o problemă puţin mai complicată – simularea deplasării unui vehicul. Vom considera vehiculul alcătuit din blocuri de caractere "□" de lungime J1 şi înălţime I1. Roţile le vom reprezenta prin " ".

Subrutina de generare a vehiculului „parcat" în punctul de coordonate (L1, K1) este:

```
5 INIT
10 FOR J=0 TO J1
20   FOR I=0 TO I1
30     PRINT AT (L1+I, K1+J); "□"
40   NEXT I
50   PRINT AT (L1+I1+1, K1+J); " ◎ "
60 NEXT J
```

Ideea de animaţie privind deplasarea vehiculului din poziţia K1 în poziţia K2 constă în ştergerea coloanei de caractere bloc din stînga vehiculului şi

adăugarea acesteia în dreapta vehiculului (v. liniile 25 şi 45 ale subrutinei de mai jos)

```
10 FOR J=K1 TO K2
20   FOR I=0 TO I1
25     PRINT AT (L1+I, J+J1+1); "□"
30     PRINT AT (L1+I, J); "  "
35   NEXT I
40   PRINT AT (L1+I1+1, J+J1+1); " ○ "
45   PRINT AT (L1+I1+1, J); "  "
50 NEXT J
60 END
```

□ Animaţie la nivel de caracter pe calculatorul AMSTRAD

vol. 2, pag. 272

Programul pe care vi-l prezentăm în continuare realizează o simulare pe calculatorul AMSTRAD privind aprovizionarea cu benzină (de trei calităţi) a trei beneficiari. El constituie un pretext pentru a putea urmări o animaţie la nivel de caracter privind: umplerea/golirea rezervoarelor şi deplasarea vehiculelor.

Lista variabilelor programului este următoarea: d (distanţa dintre centrele de simetrie ale rezervoarelor); n (numărul de rezervoare); l1, k1 (coordonatele punctului unde se generează vehiculul); t (vectorul cantităţii de benzină încărcate pe fiecare vehicul); x (vectorul absciselor centrelor de simetrie ale vehiculelor); v (matricea cu nivelul minim, nivelul maxim şi nivelul curent al benzinei pentru un rezervor k); c) matricea cantităţilor de benzină solicitate de către beneficiarul i din rezervorul j); z (vectorul ale cărui elemente indică locul de oprire al vehiculului).

Vă invităm în continuare să analizaţi singuri, fără ajutorul nostru următoarele secvenţe de instrucţiuni ale programului:

60– 160 generare rezervoare din caractere mozaic
161– 163 generare drum autovehicule
205– 235 afişare cerere benzină
240– 275 alimentare cu benzină
280– 290 „distrugere" vehicule
340– 380 subrutină de generare vehicul
390– 440 subrutină de deplasare vehicul
750– 795 simulare curgere benzină (o tonă)
950– 980 subrutină umplere rezervor
1000–1025 subrutină ştergere vehicul
1100–1135 subrutina afişare calitate (cifra octanică) benzină

A fost şi . . . penultima provocare informatică!

TEMA 13

☐ **Răspundeți prin DA sau NU la următoarele întrebări:**

– Cu instrucțiunile **PLOT** și **DRAW** putem reprezenta orice simbol de pe cărțile de joc.
– În BASIC-PRAE putem programa orice tastă pentru reprezentarea unui lup.
– În BASIC HC-85 un caracter definit se introduce în memorie cu instrucțiunea **POKE.**
– Un sprite poate fi desenat pe o grilă de 24×21 puncte.

☐ **Înlocuiți cuvintele care lipsesc din următoarele propoziții:**

a) Un sprite este o formă de reprezentare grafică de .... rezoluție pe calculatorul ....
b) Cu instrucțiunea **PRINT AT** nu se poate șterge decît ....
c) Cu instrucțiunea **PRINT AT** se poate/nu se poate șterge o linie creată cu instrucțiunea **DRAW.**
d) Pentru programarea simbolurilor grafice definite de utilizator în BASIC HC-85, TIM S, SPECTRUM procedăm astfel: a) .... , b) .... , c) ....
e) Instrucțiunea

20 **POKE USR** "A"+1, **BIN** 00011100

are următoarea semnificație ....
f) În BASIC-COMMODORE pot fi programate .... sprite-uri.
g) Prin stabilirea SPRITE POINTER-ului pentru sprite-ul #0 la adresa 255, valorile datelor pentru acest sprite trebuie citite în memorie de la adresele .... pînă la ....

☐ **Scrieți cîte un program BASIC pentru fiecare din problemele de mai jos:**

a) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC-aMIC.
b) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC-PRAE.
c) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC HC-85, TIM S, SPECTRUM.
d) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC-80.
e) Să se simuleze mișcarea unei mașini pe plan înclinat.

f) Să se simuleze mişcările unui înnotător.

g) Să se simuleze mişcările calului şi nebunului pe tabla de şah.

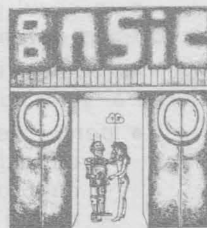
h) Să se simuleze mişcările unei sfere în aruncarea pe oblică şi pe orizontală.

i) Să se simuleze ciclul Carnot.

☐ **Să se reprezinte cu ajutorul imaginilor animate,** dinamica producţiei de jucării la Întreprinderea URSULEŢUL.

☐ **Să se reprezinte consumurile trimestriale** ale unui articol cu ajutorul imaginilor animate.

Proiectarea și realizarea unui simulator pentru o stație de livrare produse lichide. Animație, tehnica WINDOW, semnalizare optică și acustică, reprezentarea în 3D



În această ultimă conversaţie ne propunem să abordăm o problemă de grafică interactivă privind simularea pe calculatoarele personale HC-85, TIM S, SPECTRUM a livrărilor de benzină, motorină şi petrol efectuate de către o staţie de livrare produse petroliere.

Vom începe conversaţia cu o temă care permite aplicarea metodelor şi tehnicilor de grafică pe calculator.

Să se proiecteze şi realizeze un program BASIC HC-85, TIM S, SPECTRUM care să simuleze livrările de benzină, motorină şi petrol dintr-o staţie de livrare produse petroliere.

☐ Cerinţe de prelucrare

Se solicită realizarea următoarelor prelucrări:

- reprezentarea grafică (2D) a celor trei rezervoare (R_1, R_2, R_3), a pompei centrifuge şi a conductelor de alimentare;
- încărcarea rezervoarelor cu cele trei produse petroliere: benzină (R_1); motorină (R_2); petrol (R_3);
- apariţia cisternelor avînd inscripţionate numerele auto corespunzătoare (30-MS-1045; 31-BZ-3014 etc.);
- umplerea conductelor de alimentare cu produsele petroliere corespunzătoare;
- încărcarea cisternelor (în acelaşi timp cu golirea rezervoarelor corespunzătoare);
- plecarea cisternei care a alimentat şi aducerea unei alte cisterne pentru încărcat;
- afişarea orei curente (în colţul din dreapta sus al ecranului). Programul de lucru este 7–14. În momentul depăşirii orei 14 se va semnaliza (acustic şi optic) „Sfîrşitul programului de lucru". Ora curentă de începere a programului poate fi schimbată;
- umplerea conductei de alimentare va marca pornirea pompei centrifuge în vederea începerii umplerii unei cisterne sau a continuării umplerii cisternei în cazul umplerii rezervorului;
- golirea conductei va marca oprirea pompei centrifuge. Pot apare două cazuri: a) cisterna de la poziţia respectivă s-a umplut; b) rezervorul respectiv s-a golit;
- în cazul golirii unui rezervor se va întrerupe activitatea de livrare şi se va trece automat la umplerea rezervorului respectiv;
- în cazul în care o cisternă s-a umplut se va trece automat la expedierea ei şi se va aduce o nouă cisternă pentru încărcare;
- pentru fiecare din cele trei produse petroliere se vor afişa:
  - capacitatea maximă a cisternei respective (în tone);
  - cantitatea de combustibil încărcată în cisternă;
  - timpul care a mai rămas pentru încărcarea completă a cisternei;

• în funcţie de produsul solicitat, se vor genera următoarele rapoarte:
- **L1.** Lista numerelor auto şi a capacităţilor cisternelor;
- **L2.** Situaţia livrărilor faţă de planificat, în procente, pentru lunile anului în curs;
- **L3.** Situaţia valorică a livrărilor pe produs petrolier;
- **L4.** Situaţia valorică a livrărilor lunare pentru fiecare produs în parte;

• modul de lucru privind încărcarea cisternelor poate fi secvenţial (prin acţionarea unei taste) sau automat;

• livrările planificate şi realizate se vor reprezenta într-o histogramă tridimensională.

☐ Cerinţe de rezolvare

• Pentru deplasarea cisternei (intrare/ieşire) se va realiza o *animaţie la nivel de caracter.*

• Pentru descărcarea rezervorului, încărcarea cisternei, umplerea/golirea conductei de alimentare se va realiza o animaţie la nivel de pixel.

• Afişarea datelor privind starea sistemului, livrărilor etc. se va realiza prin aplicarea tehnicii *WINDOW.*

• Pentru introducerea orei şi a minutului se va realiza un *minieditor* de cifre.

• Tratarea succesivă în timp a mai multor teme (descărcare rezervor, încărcare cisternă) de acelaşi tip se va realiza prin folosirea aceluiaşi bloc de program.

• Se vor emite *semnale acustice* (avertizare acustică cu înălţimi sonore aleatoare) şi *semnale optice* (semnalizare optică intermitentă) în momentul cînd o cisternă este deja plină.

• Numerele auto şi capacităţile cisternelor se vor genera în mod aleator.

• Programul va fi ghidat de un meniu principal şi de submeniuri pentru realizarea următoarelor funcţiuni:

A – AUTO
- B – Benzină
- P – Petrol
- M – Motorină

O – OPŢIUNI
- H – Opţiune oră
- A – Livrări azi
- % – Livrări procentuale
- L – Livrări lunare
- S – Secvenţare
  - V – bucla se inchide secvenţial
  - A – bucla se inchide automat
- H – HIST

SOLUŢIA TEMEI 14

☐ Proiectarea programului

Se pot imagina numeroase variante pentru realizarea acestui simulator. În ceea ce ne priveşte, ne-am străduit să vă prezentăm o soluţie simplă în care să regăsiţi aplicate majoritatea tehnicilor utilizate curent în infografică: *reprezentare în 2D şi 3D, animaţie la nivel de pixel şi caracter, semnalizare optică şi acustică, meniuri* şi *submeniuri, tehnicile WINDOW* etc.

În fig. 14.1 este reprezentată diagrama de structură a simulatorului.

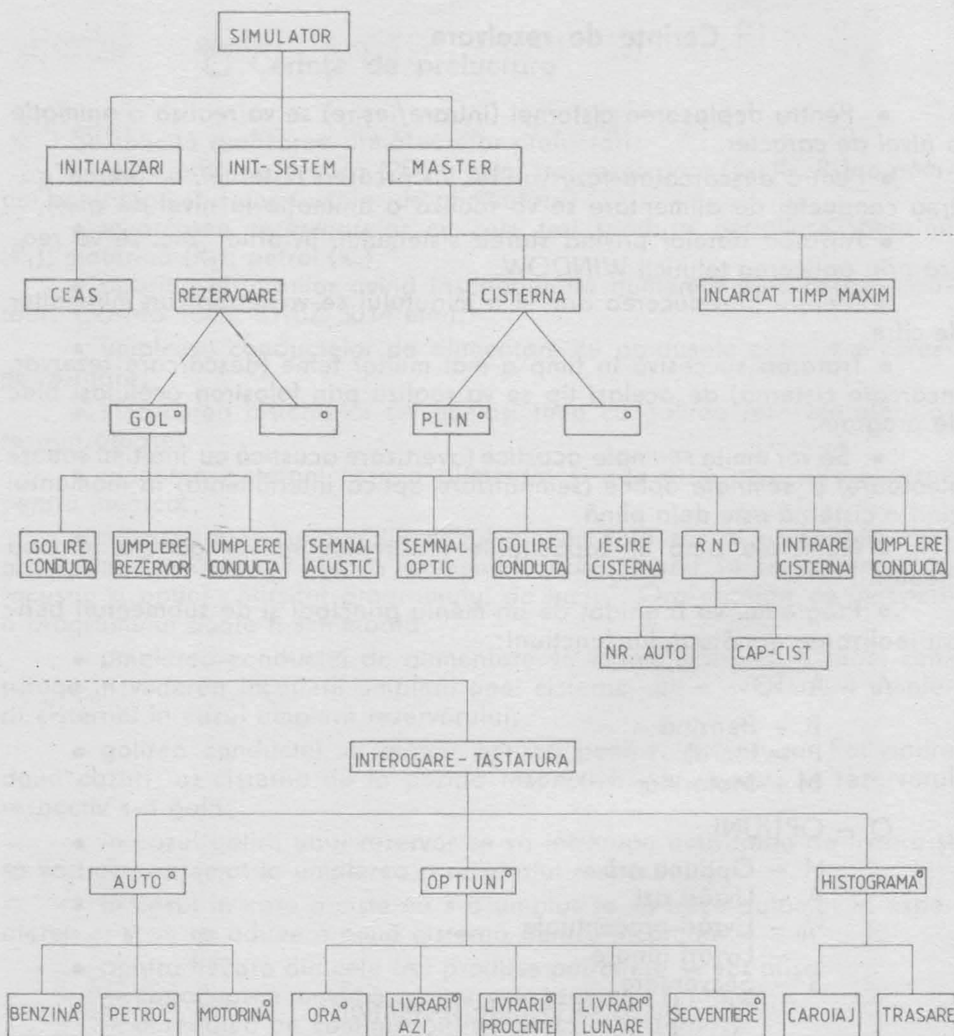


Fig. 14.1.

□ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM*

vol. 2, pag. 276

Iniţializări

Instrucţiunile 1000–1160 dimensionează tablourile de date şi le iniţializează.

Iniţializări sistem

Secvenţa de instrucţiuni 1500–1595 iniţializează variabilele simple (sum, o, m, pre, d, secv) şi indexate x(1), x(2), x(3). Se apelează subrutinele (8600, 8517, 8435, 8210) pentru umplerea rezervoarelor, alegerea capacităţii cisternelor (în mod aleator), aducerea cisternei, umplerea conductei de alimentare.

Ceas

Instrucţiunile 2105–2150 afişează ora şi minutul (incrementat cu 2) în timp real. Ceasul dă semnalul privind sfîrşitul programului de lucru.

Rezervoare

Secvenţa de instrucţiuni 2299–2324 realizează golirea rezervoarelor la nivel de pixel. Se realizează animaţia pentru umplerea/golirea conductei, umplerea rezervorului.

```
2299 REM                    – rezervoare –
2300 FOR n=1 TO 3
2310 PLOT OVER 1; 23, w (n)+3+x(n)
2315 DRAW OVER 1; 70,0
2317 LET g(n)=g(n)+1
2320 IF g(n)>=f(n) THEN LET g(n)=0 : LET x(n)=x(n)–1
2324 IF x(n)<=0 THEN GO SUB 8200 : GO SUB 8600 : GO SUB 8210
```

Cisterna

Instrucţiunile 2327–2347 realizează încărcarea cisternei pixel cu pixel. Se apelează subrutinele pentru: generarea semnalului acustic şi optic, golirea conductei, ieşirea cisternei (animaţie la nivel caracter), generarea aleatoare a numerelor auto şi a capacităţilor cisternelor, intrarea cisternelor (animaţie la nivel de caracter).

```
2327 REM                    – cisterne –
2330 LET i(n)=i(n)+.25
```

* Programul a fost realizat cu ing. Dan Drăgan, după o idee a autorului.

```
2335 IF i(n)<y(n) THEN GO TO 234
7
2337 LET y(n)=y(n)+r(n)
2340 IF INT (y(n)/(r(n)+.0001))> =3 THEN LET inc=136 : LET lung=80 : GO TO 2345
2343 LET inc=168 : LET lung=16
2345 PLOT OVER 1 ; inc, w(n)+8+INT (y(n)/(r(n)+.0001)) : DRAW OVER 1; lung, 0
2347 IF y(n)/(r(n)+0001)>16 (THEN GO SUB 8300 : PRINT FLASH 1; AT n*7-4,19;
     " ■ " : GO SUB 8200 : GO SUB 8400 : GO SUB 8517 : BEEP .04,32 : PAUSE 1 :
     BEEP .06,32 : GO SUB 8435 : GO SUB 8210 : PRINT FLASH 0; AT n*7-4,19;
     " "
```

Încărcat-timp maxim

Secvenţa de instrucţiuni 2353–2370 afişează la interval de aproximativ două secunde: cantitatea (în tone) încărcată în cisternă pentru fiecare produs (benzină, motorină, petrol); timpul (în minute) rămas pînă la umplerea completă a cisternei; capacitatea cisternei.

Interogare tastatură

Instrucţiunile 2380–2410 permit, conform meniului, alegerea uneia din următoarele opţiuni: A (auto), O (opţiuni), H (histogramă).

Auto interoghează produsul petrolier (B – benzină; M – motorină; P – petrol). În funcţie de produsul petrolier selectat se generează raportul L1.

Opţiuni cuprinde cinci alternative după cum urmează: **H** setează ora de început program (7,00 sau o altă oră din intervalul 7–14); **A** interoghează produsul petrolier (B – benzină; M – motorină; P – petrol) şi generează raportul L3; $^0/_0$ generează raportul L2; **L** generează raportul L4. **S** realizează închiderea buclei în mod secvenţial (prin acţionarea tastei V) automat (A).

Histograma 3D

Instrucţiunile 7000–7245 generează histograma tridimensională (în spaţiul 3D) a livrărilor (planificat, realizat) pentru cele trei produse petroliere: motorină, petrol, benzină. În planul orizontal se reprezintă produsele petroliere şi lunile anului: IF (ianuarie, februarie), MA (martie, aprilie) etc. Pe verticală (Z1 şi Z2) se reprezintă planificatul şi realizatul privind livrările celor trei produse petroliere. Instrucţiunile 7000–7130 realizează textul şi iniţializările pentru histogramă; instrucţiunile 7135–7180 trasează caroiajul, iar instrucţiunile 7185–7245 trasează elementele histogramei, în conformitate cu valorile lui Z1 şi Z2.

Alte subrutine

În cadrul programului mai sînt apelate subrutinele: Livrări azi (7900–8010); Opţiune ora (8100–8170); Umplerea/golirea conductei (8200–8270); Semnal acustic (8300–8340); Ieşire cisternă (8400–8432); Intrare cisternă (8435–8460); Generare aleatoare capacităţii cisterne/Generare aleatoare

numere auto (8500–8600); Umplerea rezervorului (8600–8680); Livrări procente (8700–8740); Secvențiere (8800–8840); Livrări lunare (8900–8945); 50021 (transferă imaginea binară din memoria ecran într-o zonă tampon de memorie); 50000 (aduce imaginea binară salvată cu subrutina 50021 înapoi în memoria ecran).

Imagini generate de program

Simularea livrării celor trei produse petroliere este ilustrată în fig. 14.2.

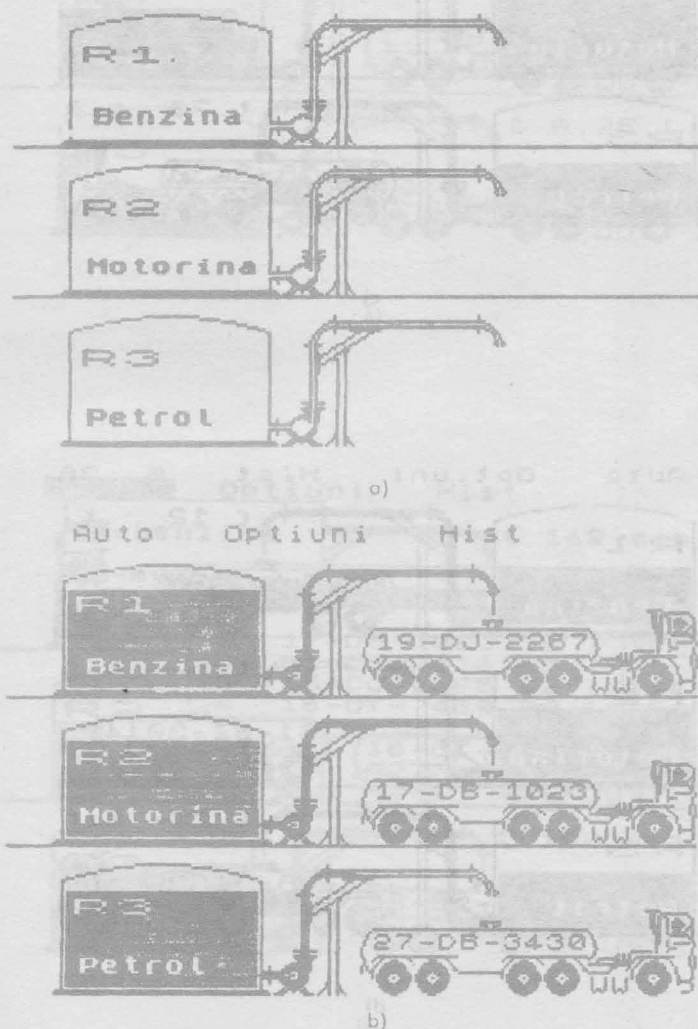
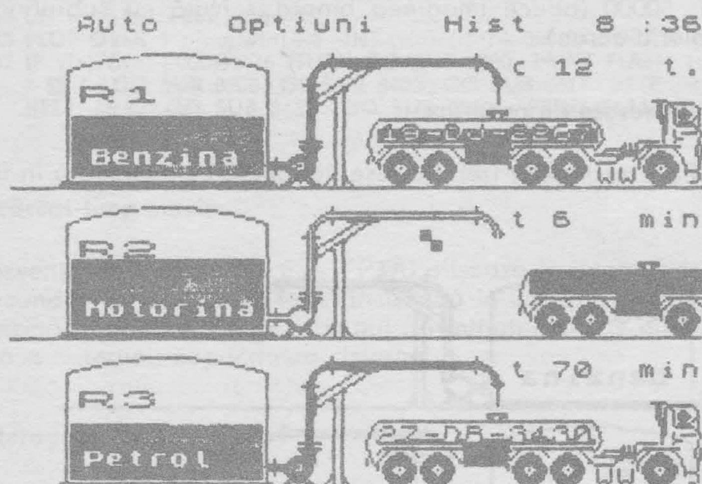
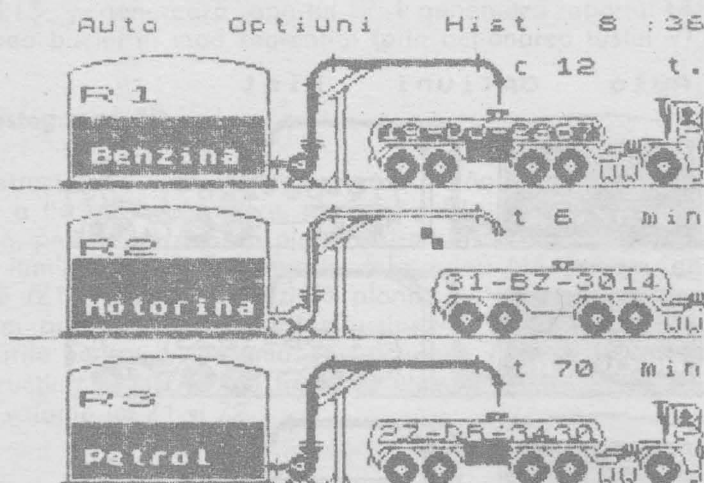


Fig. 14.2. Simulator: *a)* imaginea rezervorului înainte de inițializările sistemului; *b)* încărcarea celor trei cisterne;

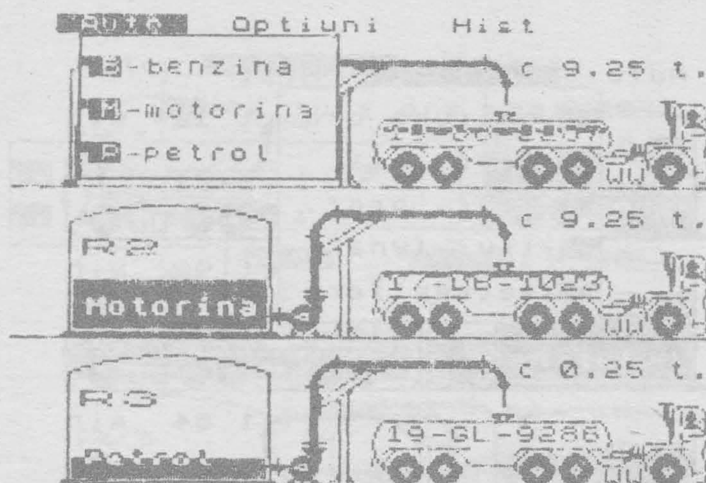


c)

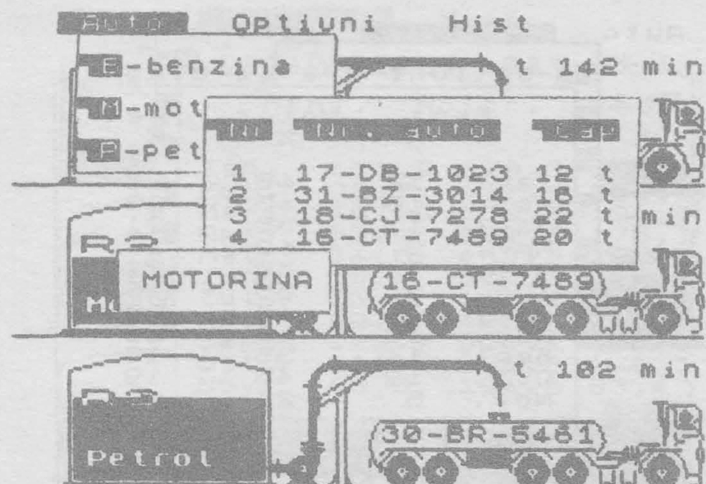


d)

Fig. 14.2. *c)* ieșirea (animată) a cisternei pline cu motorină; *d)* intrarea animată a cisternei pentru încărcarea motorinei;

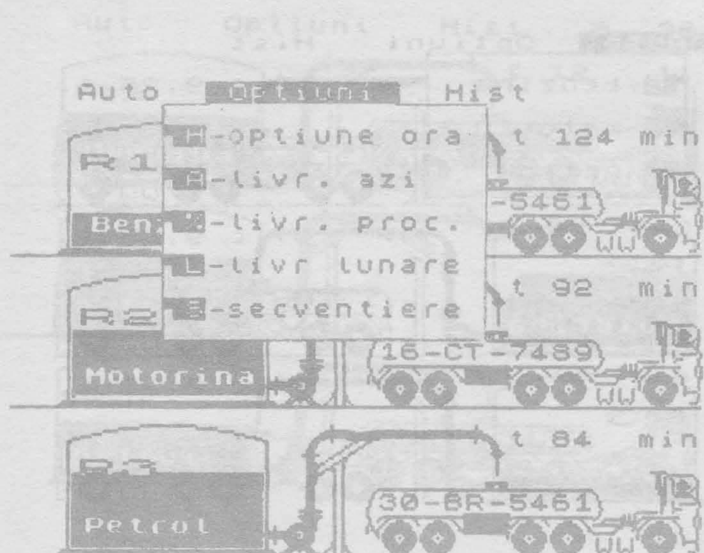


e)

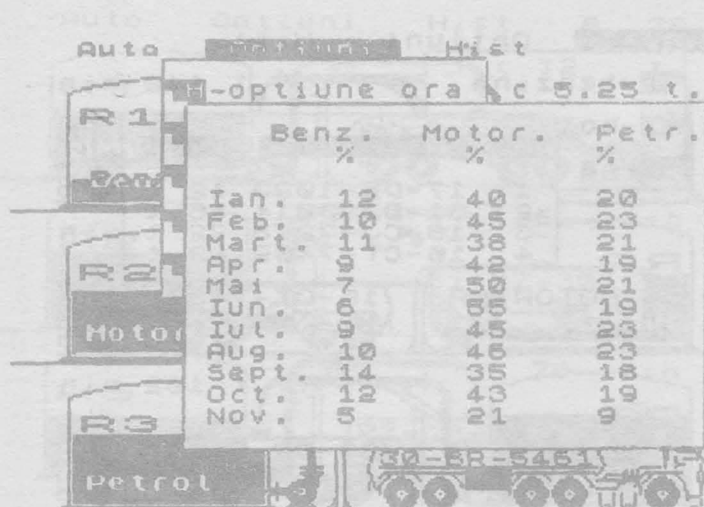


f)

Fig. 14.2. *e)* apelarea modului „Auto" din meniul principal; *f)* apelarea opţiunii „MOTO-.RINA" din submeniul corespunzător;

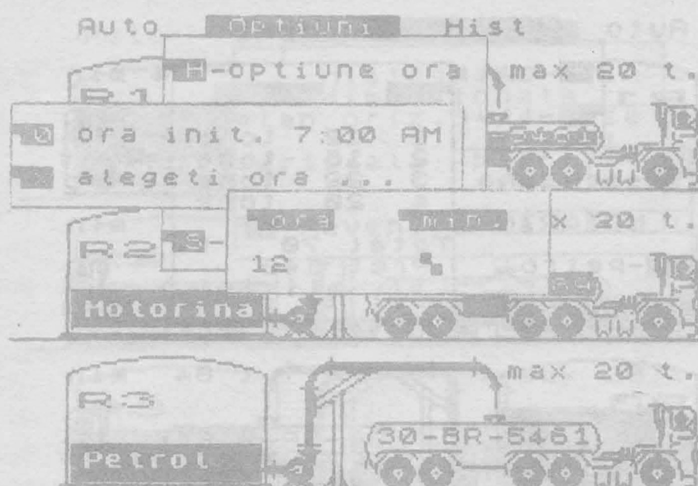


g)

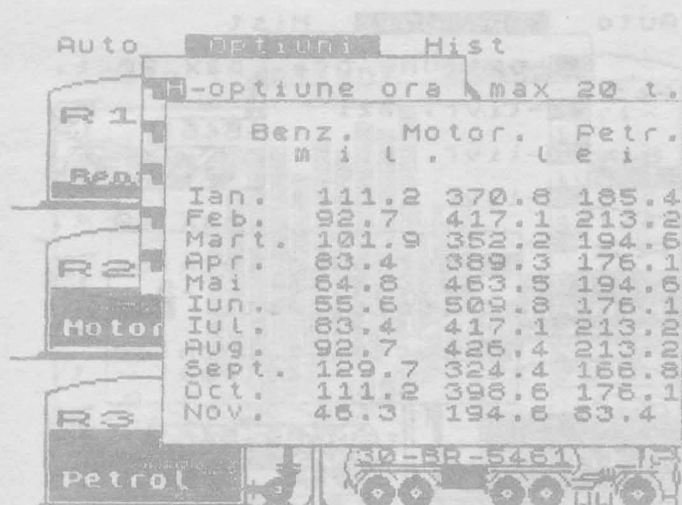


h)

Fig. 14.2. *g)* apelarea modului „opțiuni" din meniul principal; *h)* apelarea opțiunii „%" din submeniul corespunzător;

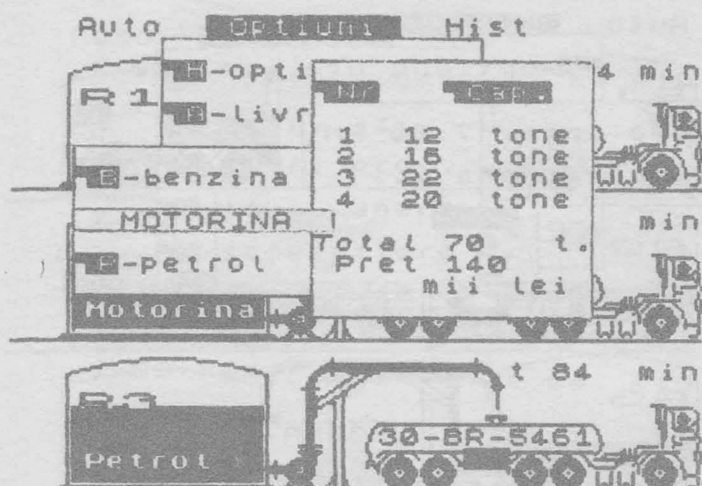


i)

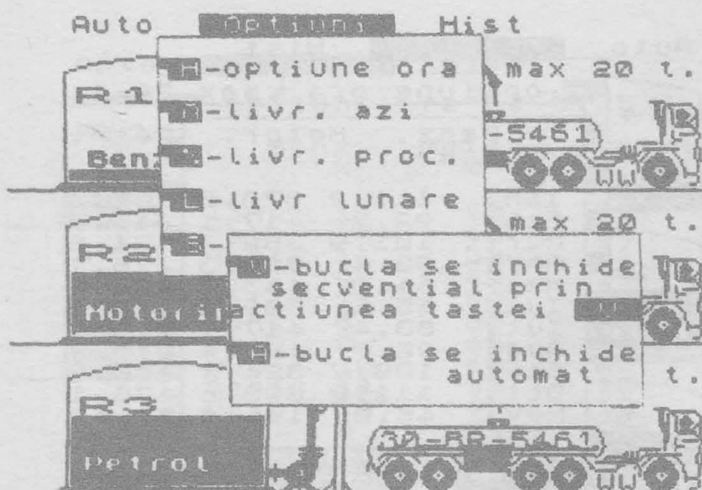


j)

Fig. 14.2. *i)* apelarea modului „Opţiune oră"; *j)* apelarea modului „L"; se afişează situaţia livrărilor lunare din anul în curs;



k)

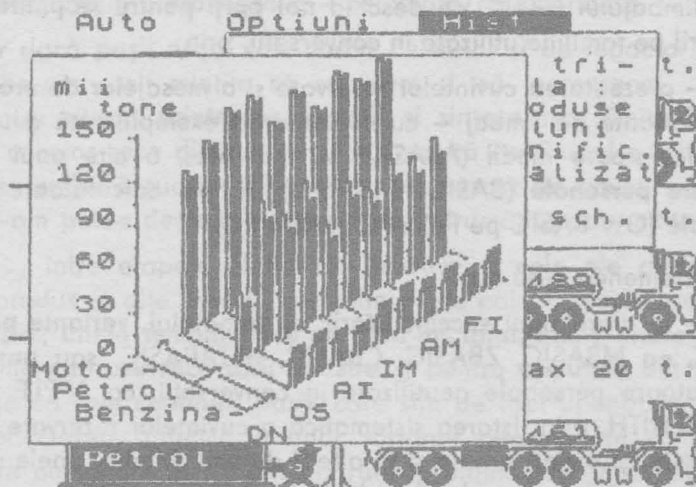


l)

Fig. 14.2. *k)* apelarea modului „A"; se afișează cantitățile livrate pînă la ora curentă, cantitatea totală livrată și valoarea livrărilor în mii lei; *l)* apelarea modului „S"; se selectează modul de lucru secvențial (V) sau automat (A);



m)



n)

Fig. 14.2. *m)* apelarea modului „H" din meniul principal; *n)* histograma 3D.



Paşi printre... conversaţii şi sinteze. Un dialog autor – editor – cititor (II)

☐ Umblînd în doi (autor şi redactor) pe 10... cărări (de vîrf)

În **conversaţii** s-a urmărit – în principal – să deprindem **cititorul** cu:

(1) – **o metodă structurată de analiză, proiectare şi programare a unor probleme simple pentru rezolvarea asistată de calculator, folosind limbajul interactiv BASIC,** implementat pe mai multe clase de calculatoare: calculatoare medii-mari, minicalculatoare, microcalculatoare, calculatoare personal-profesionale şi personale; s-au folosit felurite căi pentru "implementarea" cunoştinţelor (v. paşii de la pag. 8, 9) dar, deşi s-a prezentat pe parcurs codificarea în fiecare din cele 8 familii reprezentative de calculatoare (din care 6 familii reprezentate prin calculatoare fabricate în România), **cunoaşterea cuvintelor rezervate din variantele de BASIC nu a constituit** – cum am afirmat – **un obiectiv esenţial al conversaţiilor.**

Sintezele vin să atace (cu excepţia sintezei 20, în care sînt incluse toate programele din exemplele conversaţiilor 1–14) şi alte faţete ale cunoaşterii limbajului BASIC, ce deschid **noi porţi pentru stăpînirea practicii programării** pe familiile utilizate în conversaţii, prin:

(2) – prezentarea cuvintelor rezervate şi a mesajelor de eroare pentru diversele variante de limbaj – cu comentarea exemplificată a unui **BASIC pentru calculatoare medii** (ABASIC) şi, mai ales, a cîte unui BASIC **pe calculatoare personale** (BASIC-AMSTRAD) şi **pe calculatoare personal-profesionale** (GW BASIC pe Felix PC şi IBM PC).

De asemenea s-au dat:

(3) – în afara unui **succint istoric** al limbajului, **variante performante de BASIC,** ca MBASIC, ZBASIC, CBASIC, BETABASIC, sau **particularizări** pe calculatoare personale neutilizate în conversaţii, ca APPLE, ATARI, TI, KAPIRO, ZENITH, prin listarea sistematică a cuvintelor rezervate respective. Dar, **sintezele** îşi extind aripile şi în alte 7 direcţii diferite, unele doar "dilatînd", altele depăşind cu mult, sfera limbajului:

(4) – cea a **aplicaţiilor** limbajului în **domeniul proiectării asistate şi al reprezentărilor geometrice în BASIC,** dar mai ales în cel al **jocurilor pe lculator,** ilustrat de 25 programe sursă pentru calculatoare personale;

(5) – cea a **sistemelor de operare** (date prin caracterizări și comenzi) **tipice** pentru toate clasele de calculatoare, ca UNIX, MS-DOS, CP/M, OS/2, ca și a altora ca DOS-PC, Z-DOS, COMMODORE și APPLESOFT DOS, cu o **detaliere pentru familia CP/M,** disponibilă pentru calculatoare personale pe 8 biți și pentru calculatoare personale-profesionale pe 16 biți, fabricate în România;

(6) – cea a **microprocesoarelor** grupate pe: 8 biți (INTEL 8080, Zilog 80, MOTOROLA 6800, MOS 6502), 16 biți (INTEL 8086, Z 8000), 16/32 biți (MOTOROLA 68 000), 32 biți (80 386 INTEL) reprezentată prin caracterizări, utilizări la diverse tipuri de calculatoare și prin lista instrucțiunilor;

(7) – cea a altor **20 limbaje de programare de nivel înalt,** reprezentată prin cuvintele rezervate pentru Ada, Algol, C, COBOL, Forth, Fortran, LISP, Logo, PASCAL, Prolog ș.a.;

(8) – cea a **produselor program generalizabile** (Visicalc, Multiplan, Lotus 1–2–3, Framework ș.a.), reprezentată prin caracterizări și cuvinte rezervate, cu o extensie și o legătură cu BASIC pentru dBASE II, III, III Plus;

(9) – cum și, ca un corolar pentru limbaje, **cea integratoare a tuturor cuvintelor rezervate în 20 limbaje de programare de nivel înalt** (inclusiv în 18 variante de BASIC), ca și a celor **10 produse program generalizabile,** într-o prezentare alfabetică a cuvintelor cheie, cu indicarea pentru fiecare a limbajelor în care se utilizează;

(10) – în sfîrșit, o **bibliografie** – cercetată/recomandată.

Chiar dacă **pașii noștri** (am vorbit de autor și de redactor pentru că împreună ne-am sucit mințile să selectăm și să parcurgem... cărările optime pentru cititor) **printre conversații și sinteze** s-ar limita la această explicitare a celor **zece direcții,** din care rezultă conținutul complex al sintezelor și se reflectă succint obiectivul – cunoscut de acum – al conversațiilor, **ne-am putea declara mulțumiți** de îndrumările date cititorilor.

Dar... între etapele elaborării iterative și cele ale apariției cărții s-au mai produs și alte evoluții ale tehnicii de calcul, ale informaticii, ale limbajului ș.a., **unele semnificative pe plan mondial,** altele interesante pentru aplicațiile **caracteristice țării noastre** și pentru **opțiunile din prezentele volume.** Așa că **vom continua pașii,** care sînt de fapt **„pentru și către cititor",** cu reflectarea sintetică a unor variante performante de BASIC pe calculatoare personale de 16 biți apărute în ultimii ani (în continuarea familiilor ATARI, APPLE) ca și a unora pe calculatoare personal-profesionale recente de 16 și 32 biți (ca MACINTOSH, PS/2), cu indicarea noilor microprocesoare utilizate și a sistemelor de operare specifice, ca și a unora din extensiile hardware-software ale sistemelor anterioare, inclusiv perife-

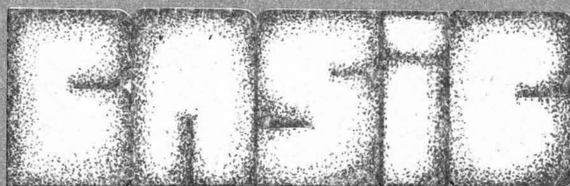
ricele şi mediul de lucru. Nu vom uita nici **clasa de superminicalculatoare** reprezentată prin elementul său de bază VAX 11 (continuatorul familiei PDP 11), care are un reprezentant fabricat de curînd în România (CORAL 8730). Vom arunca şi "ochiade" asupra tehnicii meniurilor, cum şi asupra domeniului **aplicaţiilor grafice, legat de proiectarea asistată de calculator** şi nu vom termina înainte de a mai reveni la **importanţa înţelegerii metodelor de analiză, proiectare şi programare structurată pentru asistarea cu calculatorul,** în general, şi pentru lucrul în BASIC, în particular.

Cu aceşti alţi **paşi** (v. începutul volumului 2), care continuă **dialogul*** **autor—editor—cititor,** vom considera îmbrăţişate şi actualizate toate aspectele importante ale cărţii, în... interiorul căreia se află (împovărat?!) cititorul.

* Mai am avea de spus ce... n-am spus. Pînă aici nu ne-am preocupat — şi nici nu o vom face — de calculatoarele mari, de supercalculatoare şi de eventualele limbaje BASIC ale acestora (aici este, însă, zona unor FORTRAN şi LISP performante, sau a unor limbaje specializate), limitîndu-ne plafonul la tradiţionalul Felix C, calculator mediu din generaţia a 3-a, deosebit de răspîndit în ţară, cum şi, mai ales, la mai noile mini şi microcalculatoare din generaţia 3,5→4. Celor care vor să afle ce s-a mai întîmplat în lume în drumul către generaţia a 5-a, să ştie ce e un calculator vectorial, unul „pipeline" sau unul MIMD, un masiv de multiprocesoare, în sfîrşit un **transputer** (rezultat al tehnologiilor de integrare pe scară foarte largă — VLSI — ce au culminat cu comercializarea primului microprocesor pe un singur cip conceput ca bloc de construcţie pentru calculatoarele paralele, transputerul INMOS) dar, mai ales, ce sînt limbajele paralele structurale (DAP FORTRAN, FORTRAN 8X, CMLISP) sau procesuale (OCCAM) şi ce ne... aşteaptă în viitor, nu ne rămîne decît să le recomandăm cartea **Parallel Computers 2,** de R. W. Hockney şi C. R. Jesshope, IOP Publishing LtD, Anglia—S.U.A., 1988 (aflată în curs de traducere în limba română, în aceeaşi redacţie, la Editura Tehnică).

VOL. 1+ VOL. 2: 97 LEI

TOTUL DESPRE...



..IN 14 CONVERSATII SI 7 SINTEZE

☐ Carte mult cerută, cum a reieşit din consfătuirile cu cititorii şi, mai ales, din scrisorile de la cei care lucrează sau studiază cu echipamente electronice interactive, în toate ramurile economiei naţionale.

☐ Un autor de acum cunoscut, îndeosebi prin cărţile din ciclul "Învăţăm... conversînd cu...", al Bibliotecii de Automatică, Informatică, Electronică, Management a Editurii Tehnice.

☐ Conversaţiile, ce fac obiectul primului volum, sînt orientate să asigure deprinderea cititorului cu o metodă structurată de analiză–proiectare–programare, pornindu-se de la concepţia că un program nu este bun pentru că este scris într-un anumit limbaj ci pentru că a fost bine proiectat.

☐ Se pleacă de la exemple simple, care se încheagă rapid într-o concepţie de rezolvare structurată. Documentaţii tipice (diagrame de structură, pseudocoduri, scheme logice) însoţesc ca laitmotive toate problemele.

☐ În paralel, cititorul cunoaşte codificările, comenzile şi instrucţiunile uzuale pe diferitele variante de limbaje, ce corespund diferitelor clase de echipamente româneşti sau străine.

M 118 | FELIX C

CORAL | aMIC | JUNIOR

FELIX PC | SPECTRUM | AMSTRAD

COMMODORE | INDEPENDENT | TPD

ISBN 973-31-0010-2
ISBN 973-31-0011-0